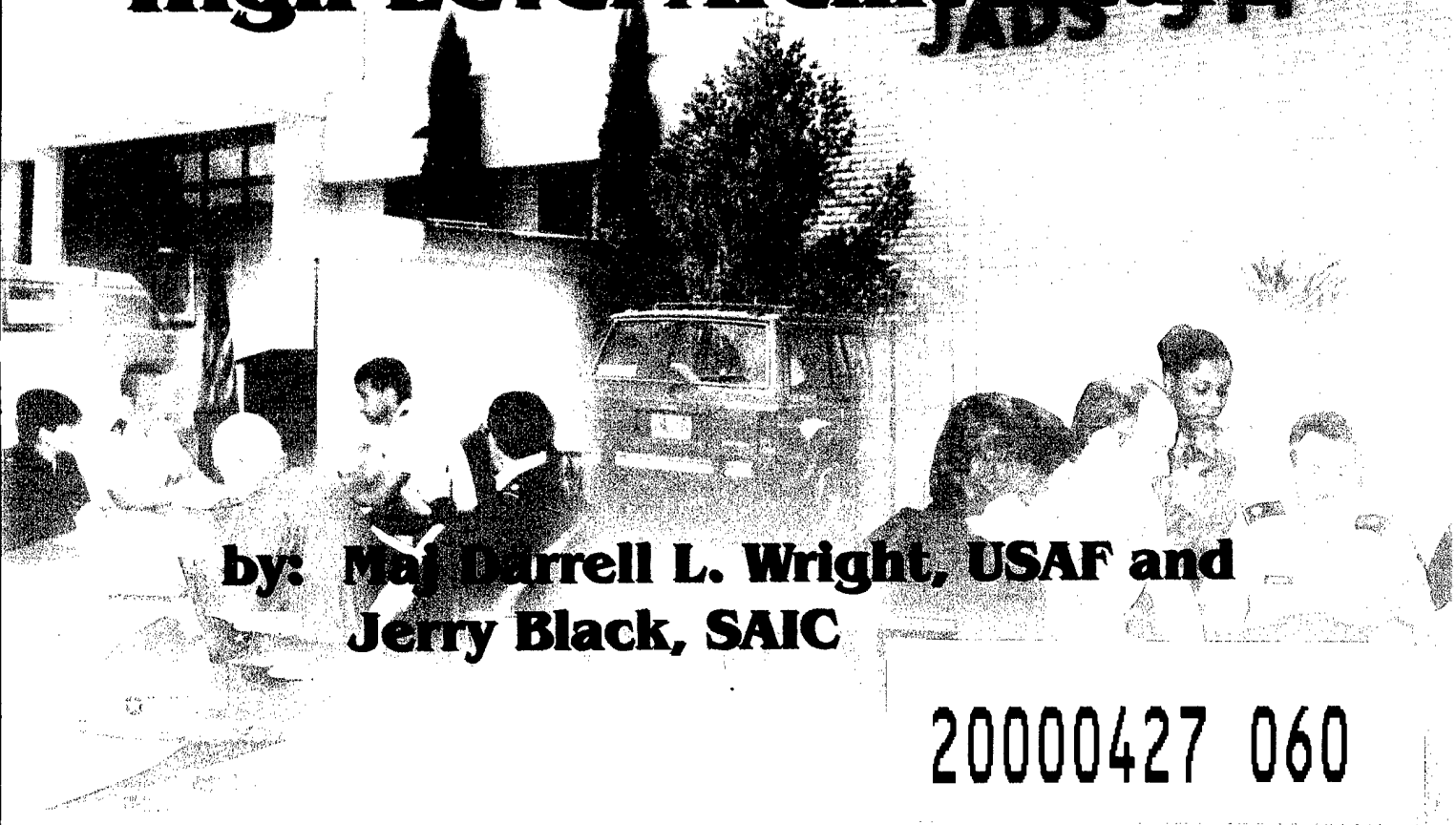
JADS JT&E-TR-00-024

JADS JT&E

JADS Special Report on High Level Architecture



by: Maj Darrell L. Wright, USAF and
Jerry Black, SAIC

January 2000

Joint Advanced Distributed Simulation
Joint Test Force
2050A 2nd St. SE
Kirtland Air Force Base, New Mexico 87117-5522

Table of Contents

Appendices

List of Figures

List of Tables

Executive Summary

1.0 Purpose

This report describes the Joint Advanced Distributed Simulation (JADS) experience with high level architecture (HLA), discusses the utility of HLA to the test and evaluation (T&E) community, and presents the requirements that HLA must meet to reach its full potential for use in T&E.

2.0 Overview

The Department of Defense (DoD) has always used rapidly evolving information systems technology to support its needs. Early efforts were sharply focused on training applications and evolved from the simulation network (SIMNET) program managed by the Advanced Research Projects Agency (ARPA) and the Army. HLA is the latest step in the effort to enable DoD simulations to connect with one another in a common virtual environment. Although it is not yet an approved Institute of Electrical and Electronics Engineers (IEEE) standard (as of the writing of this report) in 1996 Dr. Paul Kaminski, Undersecretary of Defense (Acquisition and Technology), directed DoD to make all simulations HLA compliant (Appendix B). HLA consists of an interface specification, implementation rules, and tools to help users create synthetic environments in which live, virtual, and constructive (synthetic) players can interact. The centerpiece of HLA is the runtime infrastructure (RTI) which is a distributed software application that handles all the simulation to simulation communication.

The JADS Joint Test and Evaluation (JT&E) program is an Office of the Secretary of Defense (OSD)-sponsored joint service effort designed to determine how well an emerging technology, advanced distributed simulation (ADS), can support test and evaluation activities. Because of widespread interest in using synthetic environments (and the technology and standards needed to create them) to support test and evaluation, the Air Force Operational Test and Evaluation Center (AFOTEC) felt that a JT&E program could serve as an exploratory vehicle. JADS was tasked to investigate the utility of ADS, including distributed interactive simulation (DIS) and HLA, for T&E; to identify the critical concerns, constraints, and methodologies when using ADS for T&E; and finally, to identify the requirements that must be introduced in ADS systems if they are to support a more complete T&E capability in the future

JADS investigated ADS applications in three slices of the T&E spectrum: the System Integration Test (SIT) explored ADS support of air-to-air missile testing; the End-to-End (ETE) Test investigated ADS support for command, control, communications, computers, intelligence, surveillance and reconnaissance (C4ISR) testing; and the Electronic Warfare (EW) Test examined ADS support for EW testing. The JADS Joint Test Force (JTF) was also chartered to observe or to participate at a modest level in ADS activities sponsored and conducted by other agencies in an effort to broaden conclusions developed in the three dedicated test areas.

The JADS EW Test used HLA to link manned threat simulators with a geographically separated self-protection jammer. The JADS partnership with Defense Modeling and Simulation Organization (DMSO) made the effort successful. JADS experiences with DMSO in building the EW Test architecture and in executing the test events form the basis of this report. Additional insight has come from JADS participation in the HLA Architecture Management Group and in several modeling and simulation symposia and workshops.

3.0 Key Findings

The JADS EW Test team successfully implemented HLA as part of its distributed test events. RTI performance met JADS requirements. Even though JADS experienced and solved a number of problems and even though HLA is still maturing, it is ready to be used in T&E.

HLA has utility for T&E. It is an enabling technology for distributed testing. As more simulations become HLA compliant, they become resources to the test designer looking to create a richer, more realistic synthetic environment for testing. However, as noted above, HLA is still maturing. As it matures, T&E must remain involved to ensure HLA continues to meet T&E needs and preferences. For example, T&E will favor RTI performance over adding more RTI services, while other communities may be willing to lose performance in exchange for more services. All communities will demand well-documented, high quality RTIs. However, high quality for the T&E community means more than "bug" free. T&E will also want RTIs with stable, nearly deterministic performance. T&E will find HLA more useful as it becomes more widely accepted and implemented by the modeling and simulation community at large. More HLA-based models and simulations should provide the test designer with ready resources to create richer, more realistic synthetic environments for testing. Emerging requirements for new HLA capabilities from diverse modeling and simulation communities should also be evaluated from the T&E perspective. The T&E community needs to become more educated and remain involved in HLA to ensure that HLA remains useful.

1.0 Purpose

This report describes the Joint Advanced Distributed Simulation (JADS) experience with high level architecture (HLA), discusses the utility of HLA to the test and evaluation (T&E) community, and presents the requirements that HLA must meet to reach its full potential for use in T&E.

2.0 High Level Architecture

The Department of Defense (DoD) has always used rapidly evolving information systems technology to support its needs. Early efforts were sharply focused on training applications and evolved from the simulation network (SIMNET) program managed by the Advanced Research Projects Agency (ARPA) and the Army. Conceptually, the early projects were directed toward linking training simulators with human operators at distributed geographical sites and in a common virtual environment. The players interacted with one another in this common environment in near real time. Over the years SIMNET has evolved into a technology implementation which is more flexible and robust and includes different types of simulators with different levels of fidelity. The capabilities spawned by the SIMNET evolution are now called distributed interactive simulation (DIS) and are documented in Institute of Electrical and Electronics Engineers (IEEE) Standard 1278. The high level architecture is the latest step in the effort to enable DoD simulations to connect with one another in a common virtual environment. Although it is not yet an approved IEEE standard (as of the writing of this report) in 1996 Dr. Paul Kaminski, Undersecretary of Defense (Acquisition and Technology), directed DoD to make all simulations HLA compliant (Appendix B). HLA consists of an interface specification, implementation rules, and tools to help users create synthetic environments in which live, virtual, and constructive (synthetic) players can interact. The centerpiece of HLA is the runtime infrastructure (RTI) which is a distributed software application that handles all the simulation to simulation communication.

HLA differs from its predecessors in several key aspects. It is based on object-oriented design concepts. The terminology, recommended design processes, programming language of choice (C++), and tools of HLA borrow heavily from this software design methodology. However, while the architecture uses the terminology, it does not completely implement all the concepts of object-oriented design. Those unfamiliar with object-oriented design can be easily lost in the jargon and design approach. Those very familiar with object-oriented design may assume more capabilities than are actually implemented. We highly recommend attending Defense Modeling and Simulation Organization (DMSO)-sponsored training to become familiar with the terms, concepts and specifics of the implementation. The remainder of this report is written using HLA terms. The following paragraphs are a brief description of the key concepts and terms used in HLA. We have also compiled a list of common HLA terms with their definitions (Appendix A).

2.1 HLA Overview

This section was developed directly from a presentation made by Dr. Judith Dahmann, DMSO, at the Spring 1997 Simulation Interoperability Workshop. The entire presentation can be downloaded from the Simulation Interoperability Standards Organization (SISO) Web page at http://siso.sc.ist.ucf.edu/.

HLA was created in response to shortfalls in its predecessors, DIS and aggregate level simulation protocol (ALSP). DIS allows users to create a synthetic environment within which humans may interact through simulation(s) at multiple sites networked using compliant (IEEE Standard 1278-1, 1278-1A, and 1278-2) architecture, protocols, standards, and databases. DIS was created to meet the needs of the real-time, platform-level niche of the modeling and simulation (M&S) market. DIS used fixed message structures and messages were broadcast to all players. On the other hand, ALSP was created to meet the needs of the discrete-event, logical-time niche of the M&S market. It was designed to accommodate legacy simulations. Both DIS and ALSP were limited and neither provided a single technical architecture for distributed M&S. HLA was created to answer the needs of both the DIS and ALSP communities as well as provide a bridge for each by being the single technical architecture.

HLA was built on the following premises.

- No single monolithic simulation can satisfy the needs of all users.
- All uses of simulations and useful ways of combining them cannot be anticipated in advance.
- Future technological capabilities and a variety of operating configurations must be accommodated.

As a result, HLA was created to allow simulations, live entity surrogates, viewers, and data collectors to interact with one another by separating the functionality of each from the general purpose supporting runtime infrastructure. (In HLA terms, each simulation, live entity surrogate, viewer, data collector is called a federate. A federation is a named set of interacting federates.) In order to interact, the federation of simulations has to have a common understanding of player relationships and interface. HLA provides both. The architecture specifies the following.

- Ten basic rules that define the responsibilities and relationships among the components of the federation.
- An object model template that specifies the form in which simulation elements are described.
- A runtime interface specification that describes the ways that simulations interact during an operation. (This specification allows the simulations to interface with software called the runtime infrastructure. The RTI is essentially a distributed application that provides the simulation to simulation communications, provides time management services, and performs other federation control functions. Each simulation deals with a local RTI instance.)

HLA is really a standard that does not mandate a specific software implementation. (This is an important concept that will get explored later.) The rules are a high-level articulation of responsibilities of each federate.

1. Federations shall have an HLA federation object model (FOM) documented in accordance with the HLA Object Model Template (OMT). (A FOM is similar to a software interface specification.)
2. In a federation, all object representations shall be in the federates, not in the RTI.
3. During a federation execution, all exchange of FOM data among federates shall occur via the RTI.
4. During a federation execution, federates shall interact with the RTI in accordance with the HLA interface specification.
5. During a federation execution, an attribute of an instance of an object shall be owned by only one federate at any given time.
6. Federates shall have an HLA simulation object model (SOM) documented in accordance with the HLA OMT. (This rule requires that each simulation describes the functionality it is able to provide to a federation in OMT terms. All functions may not be used in any given federation.)
7. Federates shall be able to update and/or reflect any attributes of objects in their SOM and send and/or receive any SOM object interactions externally, as specified in their SOM.
8. Federates shall be able to transfer and/or accept ownership of attributes dynamically during a federation execution, as specified in their SOM.
9. Federates shall be able to vary the conditions (e.g., thresholds) under which they provide updates of attributes of objects, as specified in their SOM.
10. Federates shall be able to manage local time in a way which will allow them to coordinate data exchange with other members of a federation. (Simulations in a federation must manage time so that there appears to be one clock. Internally, a simulation manages time any way it wishes, as long as it meets its commitments to other simulations in the federation.)

There are two key HLA elements that appear several times in the rules. The first is the OMT. The OMT provides a standard format for specifying the capabilities and interface requirements of a federate via a SOM or a federation via a FOM. DMSO provides a tool, the object model development tool (OMDT), to make it easier for users to create the FOMs and SOMs called for in the rules. (JADS experience with the OMDT is described in Section 12.5.1.) Simulation owners who want their simulations to be certified HLA compliant will need to obtain the OMDT and get acquainted with it as they prepare their SOM.

The second key element is the RTI. As noted above, the RTI is a distributed application that provides the backplane that allows simulation to simulation communication, provides federation management services, and when required, time management services. Simulations interface with the local RTI component. The local RTI components communicate with one another to move the data around and to perform the management services. The interface specification standardizes the RTI/simulation interface as well as sets some limits on RTI behaviors. However, the interface specification does not require any particular software implementation. While DMSO has created several RTI versions to prove the interface specification, commercial RTI versions

will become the norm in the future. Just prior to the creation of this report, the first commercial RTI version RTI 1.3 Next Generation was certified.

2.2 Implications of HLA

HLA places no requirements on data format or meaning. There are no fixed message structures replacing the DIS standard. This allows the designer more flexibility in tailoring the messages to fit the problem at hand and to more easily live within constraints of the network architecture. However, it increases the complexity of integration. The burden of integration under HLA rests entirely on the designers of the federation to agree on what objects will be present in the federation, how the objects will be represented, what federate owns each object, what object attributes are needed by other objects in the scenario, and what interactions will occur among objects. Within HLA, the common understanding is captured in the FOM. Another tool, called the Federation Execution Planner's Workbook (FEPW), is used to map the objects and data messages onto the communications/computer hardware elements. Users may find these need to be augmented to fully capture the complexity of the federation.

HLA provides a common interface to the communications infrastructure. This opens the door for the simulation to interact with other simulations. The HLA interface specification addresses the interface between the simulation and the RTI. This allows the simulation designer to treat the communications with other federates in a more abstract manner by just calling RTI services. The HLA places no requirements on network protocols or RTI implementation details beyond those contained in the interface specification currently being considered by IEEE. Likewise, current RTIs do not interoperate, and there is no requirement in the interface specification to make them interoperate. This allows room for commercial RTI developers to create better RTIs by applying the latest technology and network protocols. It also effectively isolates the simulation from the network protocols. This in turn gives the federation the flexibility to select the RTI that best satisfies the federation requirements. Simulations that expect to operate within several different federations need to provide an interface that allows different FOMs and their associated message structures to be changed out with minimal changes to the simulation. This is sometimes referred to as a flexible interface.

The interface specification allows two different levels of message delivery service: best effort and reliable. Current RTI implementations use user datagram protocol (UDP) multicast for best effort and transmission control protocol (TCP)/Internet protocol (IP) for reliable. While these are standard protocols, implementation in operating systems and communications hardware varies. This ultimately impacts the performance of the HLA federation and sets the limits on message throughput and latency. Attempts to reduce latency or improve throughput must address the host computer operating system, protocol implementation, RTI performance parameter settings, local area network implementation, and wide area network implementation (if required).

The federates themselves declare what they can provide (publish) to the federation and what they want from the federation (subscribe). The idea is to have the simulation in the federate deal only with data it requires or wants. The only guarantee that HLA provides is that the federate won't have to deal with unwanted messages. There is a temptation to believe this will save bandwidth.

It is up to the RTI developer to decide if filtering will be done before the data are transmitted or after they are received by the local RTI component.

The initial implementations of the RTI tried to capitalize on an emerging Object Management Group standard. The Common Object Request Broker Architecture (CORBA) is the Object Management Group's answer to the need for interoperability among the rapidly proliferating number of hardware and software products available today. Simply stated, CORBA allows applications to communicate with one another no matter where they are located or who has designed them. (www.omg.org/corba/whatiscorba.html) Even though CORBA was created with client-server software applications in mind, it has features that make it attractive to use in an RTI. While current RTI versions have moved away from CORBA, it may well find its way back into future versions. The use of CORBA may have implications on RTI performance. Federation designers should not treat the RTI as a black box.

3.0 JADS Overview

The JADS Joint Test and Evaluation (JT&E) program is an Office of the Secretary of Defense (OSD)-sponsored joint service effort designed to determine how well an emerging technology, advanced distributed simulation (ADS), can support test and evaluation (T&E) activities. Because of widespread interest in using synthetic environments (and the technology and standards needed to create them) to support T&E, the Air Force Operational Test and Evaluation Center (AFOTEC) felt that a JT&E program could serve as an exploratory vehicle. Both the developmental and operational test communities shared interest. Accordingly, the JADS JT&E program was nominated. The services concurred in the need for rigorous examination of the use of synthetic environment technology and the OSD Deputy Director of Test and Evaluation chartered JADS as a joint test program in October 1994. JADS was chartered to investigate the utility of ADS for both developmental test and evaluation (DT&E) and operational test and evaluation (OT&E). JADS was tasked to investigate the utility of ADS, including DIS and HLA, for T&E; to identify the critical concerns, constraints, and methodologies when using ADS for T&E; and finally, to identify the requirements that must be introduced in ADS systems if they are to support a more complete T&E capability in the future.

JADS investigated ADS applications in three slices of the T&E spectrum: the System Integration Test (SIT) explored ADS support of air-to-air missile testing; the End-to-End (ETE) Test investigated ADS support for command, control, communications, computers, intelligence, surveillance and reconnaissance (C4ISR) testing; and the Electronic Warfare (EW) Test examined ADS support for EW testing. The JADS Joint Test Force (JTF) was also chartered to observe or to participate at a modest level in ADS activities sponsored and conducted by other agencies in an effort to broaden conclusions developed in the three dedicated test areas.

4.0 JADS EW Test

4.1 EW Test Description

To determine the utility of ADS technology for EW T&E, JADS used the HLA in two phases of the three-phased test program. The test phases were designed to allow the direct statistical comparison of ADS-based test results to results from traditional tests to measure the impact of ADS. To accomplish this, a reference test condition (RTC) was established and recreated in each test phase. The RTC defined the aircraft flight profile, the threat engagement zones and rules, the self-protection jammer (SPJ) pod responses, and the relevant data to be collected. Phase 1 was a series of traditional tests accomplished on an instrumented open air range (OAR) and in a hardware-in-the-loop (HITL) facility. In both environments, JADS used similar manned threat simulators, the same SPJ pod, and the same aircraft flight profiles. The HITL tests used time-space-position information (TSPI) recorded on the OAR to remove potential sources of variation. The radio frequency (RF) environment, the threat systems, and the SPJ were all instrumented to calculate standard EW measures of performance from the data collected. Phase 2 and Phase 3 used HLA to link different representations of the jammer to the HITL facility threats. (Phase 2 used a real-time digital system model while Phase 3 used the actual jammer mounted on an F-16 suspended in an anechoic chamber.) Other elements necessary to recreate the OAR environment (such as aircraft position and attitude, threat simulator activation/deactivation commands, and other RF emissions activation/deactivation) were represented by simple federates that played data recorded in the OAR. These federates were brought together in a federation to gather data to evaluate the utility of ADS. The federate interactions were monitored, and the measures of performance were calculated in real time.

Because JADS used HLA in developing the ADS architecture, the key operating component supporting the JADS test federations was the RTI. JADS also attempted to use the available tools and recommended processes to determine how well the HLA would support the test and evaluation community. The direct comparison of traditional and ADS-based test results caused JADS to instrument the architecture to attempt to isolate where ADS-induced errors might appear. This resulted in a rigorous examination of the architecture including the RTI.

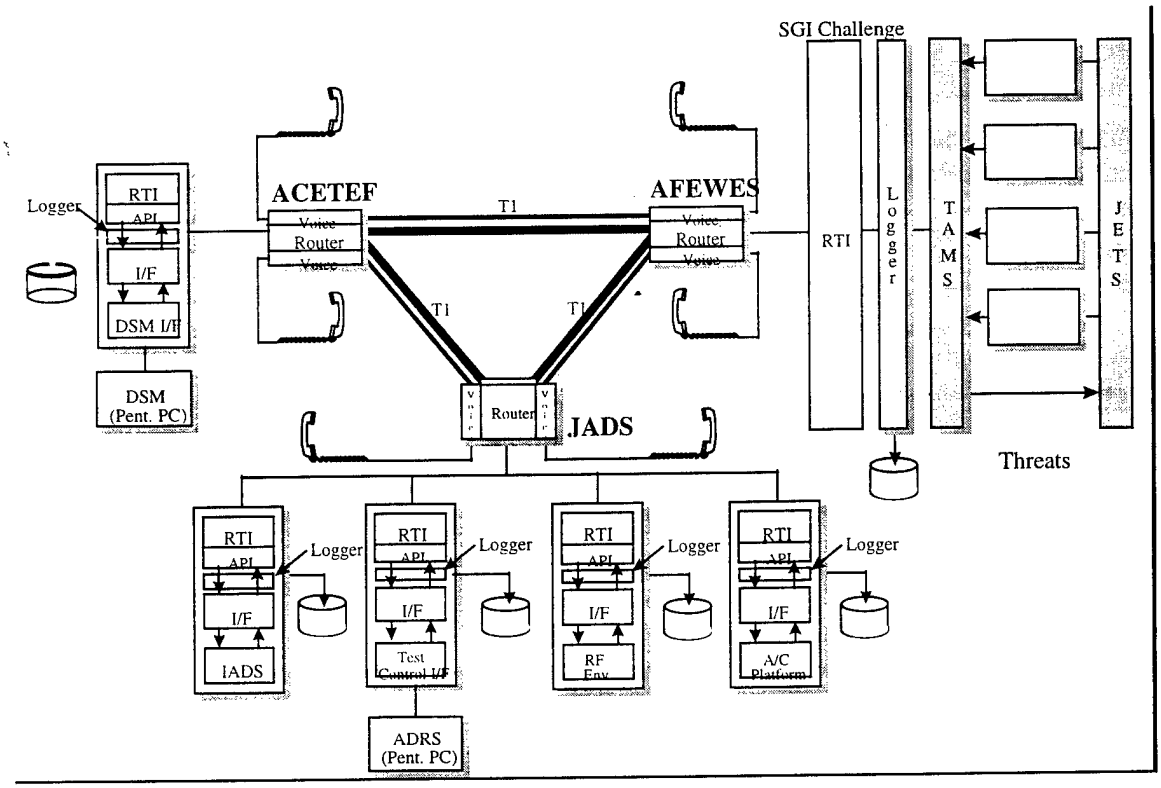
4.2 EW Test Requirements

The problem space was defined by the RTC used in the OAR test. Closed-loop testing using ADS technology runs the risk that the communications infrastructure transmitting the data interchange between objects (federates in this case) will affect the outcome either through lost interchanges or by altering the temporal nature of the interchange. This temporal change is usually an increase in the time for the interchange to occur and is called latency. The amount of allowable latency depends on the nature of the interchanges and the decision cycle of each system involved. The EW Test interchange of interest was the threat radar activation, jammer identification and response, and associated threat response.

We focused on determining how much latency the jammer/threat interaction could tolerate and still be a valid test. Depending on how the engagement is carried out, the interaction can be the jammer's computer working against the threat's computer or the jammer's computer working against the threat's human operator. Latency is driven by the decision cycle times of the jammer computer and either the threat computer or the threat operator. The jammer used in the JADS test was simple and had a very short decision cycle. Likewise the threat computers had very short decision cycles. The analysis showed that it was unrealistic to model the computer to computer interaction. The latency expected from linking the HITL manned threat simulators at the Air Force Electronic Warfare Evaluation Simulator (AFEWES) in Fort Worth, Texas, to the jammer representation located at the Air Combat Environment Test and Evaluation Facility (ACETEF) at Patuxent River, Maryland, was too great to faithfully reproduce the engagements that normally occur at distances shorter than 50 kilometers (km). In fact, the analysis indicated that once the wide area network (WAN) communications time, the local area network (LAN) communications time, and the facility interface processing times for both AFEWES and ACETEF were accounted for, the acceptable latency for the RTI had to be a negative value. The decision cycle time for the threat operator was estimated to be 500 milliseconds (ms), which we believed was an achievable latency objective for JADS. Therefore, the limitation that we placed on the communication infrastructure latency with human operator interaction was 500 ms. Once the total latency was identified, the 500 ms were allocated to the communications infrastructure, facility interfaces, and the RTI. That means that from the time the threat radar changed state, the infrastructure had no more than 500 ms to get that message to the jammer and then return the jammer's response. (The jammer's decision cycle time or "response time" was not included in the 500 ms.) We referred to this as an "end-to-end interaction" during the EW Test.

4.3 Network and Hardware Description

The EW Test used dedicated T-1 circuits, communications, and encryption devices to link JADS with two key EW facilities, AFEWES and ACETEF. Three network nodes interconnected a total of seven federates. Five federates represented critical components of the OAR test environment including the test aircraft, aircraft EW systems, and threat systems. Two federates provided a test control and test analysis capability. The JADS test control facility at Albuquerque, New Mexico, hosted four of the seven federates executing on dedicated Silicon Graphics, Inc., (SGI) O2 and a fifth federate executing on a Sun SPARCstation. (The federate hosted on the Sun SPARCstation was a data logger and visualization tool. It was a late addition into the architecture to reduce risk in test control and analysis. It only subscribed to data. When it functioned properly, it had no measurable impact on the federation. It is not in Figure 1, nor is it discussed any further in this report.) There was one federate executing on an SGI O2 at the ACETEF and one federate executing on an SGI Challenge at the AFEWES HITL facility. The Phase 2 network architecture is illustrated in Figure 1.



A/C = aircraft
ADRS = Automated Data Reduction Software
API = application program interface
DSM = digital system model
Env = environment
I/F = interface
IADS = Integrated Air Defense System
JETS = JammEr Techniques Simulator
PC = personal computer
Pent = pentium
TAMS = Tactical Air Mission Simulator
T-1 = digital carrier used to transmit a formatted digital signal at 1.544 megabits per second

Figure 1. JADS EW Test Phase 2 Test Architecture and Federates

JADS added network sniffers into the Phase 3 architecture to provide more insight into data loss problems seen in the Phase 2 execution. One sniffer was added at each site in parallel to the LAN to record all incoming and outgoing WAN message traffic. The data loss problems were not seen in Phase 3. However, these sniffers allowed us to examine other phenomena and were essential in discovering the single instance of a reliable message being lost during the Phase 3 data analysis.

RTI time management services were not used. Time synchronization came from an Inter-Range Instrumentation Group (IRIG) B signal provided at each facility by a local global positioning system (GPS) receiver. All federate computers had a card installed that could use the IRIG B signal as a time source. This proved generally acceptable, although we found federates that read time directly off the card were more accurate than federates that used system time and additional software to set the system time using the time card as the reference.

JADS message structures and data logging scheme were designed to aid in the analysis of latency. JADS required each federate to add a time stamp when the message was created. The data logger JADS created logged all data at the simulation/RTI application interface at each federate. The logger time stamped and logged each message as it crossed the interface. JADS

also required that each message from each object be numbered sequentially by the federate. The combination of the time stamping at several points in the transmission chain from data production to data consumption coupled with the synchronized clocks, the unique message number, and the network sniffers (in Phase 3) allowed JADS to track each message through the entire network. JADS could also quickly identify any lost messages and pinpoint where the message was lost.

JADS constructed a WAN test bed to reduce the WAN/LAN/RTI integration risk for Phase 2. The test bed was located at JADS in Albuquerque, New Mexico. It was constructed from the WAN hardware prior to activation of the T-1 lines and installation of the WAN hardware at the two remote sites. After Phase 3 was complete, the network was shut down and the WAN hardware returned to JADS. JADS reassembled a significant portion of the test bed to examine other RTI issues. These results are discussed in sections 5, 7, and 9 in this report.

4.4 JADS EW Test Federation

The JADS EW Test federation (the federation) linked seven federates passing attributes and interactions (messages) within constrained timing tolerances representing EW systems operating in a live test environment. A federate residing at ACETEF in Patuxent River, Maryland, represented the jammer. The surface-to-air missiles (threats) were represented by a federate at the AFEWES facility in Fort Worth, Texas. The AFEWES federate represented two threats during a normal execution of the JADS EW Test. Two additional threats were represented by the radio frequency environment (RFENV) federate. Five federates were located in the JADS Test Control and Analysis Center (TCAC) facility in Albuquerque, New Mexico. The terminal threat hand-off federate (HANDOFF) cued the threat operators. The test control federate (TCF) controlled the start/stop of the federation and passed data to the real-time analysis tool (Automated Data Reduction Software [ADRS]) executing on a personal computer (PC). The RFENV federate published RF background. The platform federate (PLATFORM) published aircraft TSPI. The analysis federate subscribed to all data published by the other federates and provided a real-time display of the engagement.

The federates joined the federation one at a time starting with the federates residing in the TCAC. As a federate joined, it began publishing link health check (LHC) messages. As soon as all federates had joined and other equipment was ready, TCF sent a start message to begin the run. After the start was received, all playback federates began publishing the data in their script. When a threat detected the aircraft, the AFEWES federate published mode change interactions as the threat's mode changed. Upon receipt of the mode change, the jammer federate published a jammer response interaction. At some time during the engagement, the threat fired missiles at the aircraft, and the AFEWES federate published missile entity state position attributes. After the run was complete, TCF sent a stop message, all federates resigned and the federation was destroyed. Each run (start to stop) lasted approximately four minutes. For Phase 2 of the JADS EW Test, the federates used RTI 1.3 release 4. RTI 1.3 release 5 was used for Phase 3. Table 1 shows the maximum observed messages published per second for each federate.

Table 1. Maximum Publish Rates by Federate

Federate	Maximum Messages Published (per second)
ACETEF	28
AFEWES	116
HANDOFF	3
PLATFORM	43
RFENV	3
TCF	2

5.0 RTI Performance Tests

The federation development and execution process (FEDEP) recommends integration testing be accomplished prior to execution. JADS concerns led to the early creation of a test bed comprised of the actual communications hardware and computers that would be used in the EW Test. This provided JADS with the capability to test and tune all the components prior to the creation of the WAN. One of the key elements was the RTI. JADS tested several RTI versions in the test bed prior to executing Phase 2. These included one version of RTI 1.0; two prerelease versions of RTI 1.3; and three post-release versions of RTI 1.3; release 3, release 4, and release 5.

During the course of test build-up and execution, JADS learned that the test bed was a unique capability. After Phase 3 was complete and the WAN equipment was returned, we reassembled the test bed to examine RTI 1.3 release 6 and two commercial RTIs, RTI 1.3 NG (beta) and a reduced service RTI (1.3.1b) from Mak Technologies. Since there wasn't enough time for tuning the architecture as JADS had done with the earlier RTI versions, JADS ran the RTIs using the default settings. Users should be able to get somewhat better performance from each by tuning the architecture. The RTI 1.3 NG (beta) is not the version that DMSO recently certified as compliant with the 1.3 interface specification. The released RTI 1.3 NG should perform better than the beta version.

This section discusses the test methods and results of both the pre-Phase 2 and post-Phase 3 testing. Results are presented for RTI 1.3 release 4, release 5, release 6, RTI 1.3 NG beta, and Mak RTI 1.3.1b.

5.1 Pre-Phase 2 Test Objective

The primary objective of JADS RTI testing was to ensure that the EW Test had an acceptable communications infrastructure, including the RTI, for each ADS test phase in order to accurately recreate the critical interactions from the OAR test environment. Acceptable meant that all hardware and software components were behaving as required and that the total system latency was within budget over the expected range of message rates and sizes used to recreate the OAR test event interactions.

Prior to Phase 2, JADS conducted RTI tests to satisfy two key requirements.

- Quantitatively measure latency and expected RTI 1.3 software performance prior to JADS EW Test Phase 2 and Phase 3.
- Provide input to the verification, validation, and accreditation (VV&A) process for JADS EW Test Phase 2 and Phase 3.

JADS conducted two types of RTI tests to meet these requirements. One-way testing was conducted to measure raw performance of the RTI between a sender and a receiver. Multiple federate testing was conducted to predict the performance of the JADS federation.

RTI test results were provided on a regular basis to the DMSO technical support and RTI software developers.

After JADS completed Phase 3, we had time to briefly examine the performance of the latest RTI versions. Since we froze our Phase 3 configuration with RTI 1.3 version 5, we had no experience with several RTI versions. The test bed was to transition to the Foundation Initiative 2010 project, so JADS needed to determine if the tools and test bench could be used to test other RTIs. There was also interest within the community for JADS to test the latest RTI versions on the JADS test bed. JADS reassembled the test bed and performed basic tests on RTI 1.3 release 6, RTI 1.3 NG (beta), or the Mak RTI. The results presented in Section 5.3 were gathered after we completed Phase 3. We had a limited amount of time to collect the data. As a result, only a single execution of the test matrix was run with each of the RTI versions. The results are included in this report to illustrate the types of results we obtained. They are not a statistically significant sample. Where we had data for a particular RTI, we compared our results with our previous results to provide ourselves confidence that the test bed was operating correctly.

5.2 Test Software

The following is a description of the software tools JADS developed for testing the network architecture (including the RTI) in 1998. DMSO has since developed its own benchmarking tools. JADS has no experience with the DMSO tools, so they are not discussed.

There are two types of software JADS developed for the RTI tests -- one-way software (non-RTI and RTI versions) and runtime configurable test federate software. Using the JADS test bed configuration, the one-way tests characterized the network and the RTI in the simplest of cases. The non-RTI tests gave an approximation of the raw network performance. There are three one-way test tools. The first used IP multicast without the RTI. The second used TCP/IP without the RTI. The third used the RTI and could be configured for either best effort (IP multicast) or reliable (TCP/IP). All three tools sent messages from the sender to the receiver. The size and the frequency of the messages changed until the test matrix was complete. The one-way RTI software indicated the performance of the RTI and network combined. By comparing the RTI results with the raw network results, the performance of the RTI could be approximated. By

modifying the federation execution data (FED) file, the tests could be executed using reliable (TCP) or best effort (IP multicast) transmission.

The second type of software developed was runtime configurable test federate software (*testfed*). The *testfed* is an RTI federate capable of running in different configurations on multiple computers within a federation execution. The purpose of this software was to determine how the RTI performed in a more realistic environment under loads anticipated for the JADS federations. The *testfed* federate accepts command line arguments that specify the characteristics of an instance of the federate. The user can specify the federate identification (ID) number (-f), the duration of the test (-d), the size of the attributes and interactions (-s), the rate that attributes are published (-r), the number of updates at the specified rate (-n), the amount of time the federate should wait before starting to publish at its specified rate (-w), and whether interactions should be published (-i). There are only one attribute and one interaction used by all federates. All federates subscribe to the attribute and the interaction.

5.3 One-Way Test

5.3.1 Description

The one-way tests were designed to exercise a communications link and the RTI with different data sizes and transmit rates. The sizes varied among 17, 51, 101, 301, 501, and 1001 bytes. The transmit rate varied among 5, 10, 20, 50, 100, 200, 400, and 500 hertz (Hz). The complete matrix of rate and size combinations was tested. Each test case, which consisted of a rate and size pair, ran for two minutes. A separate matrix test was executed for TCP, IP multicast, RTI reliable, and RTI best effort data.

5.3.2 One-Way Test Results

Table 2 summarizes the performance of IP multicast and RTI best effort transmission during the one-way tests. In the tests with RTI 1.3r5, 13% of the published packets were lost when publishing 301 bytes at 400 Hz. The maximum loss (72%) occurred when publishing 1001 bytes at 500 Hz. These results represent a single pass through the test matrix. The results for RTI 1.3r4 and RTI 1.3r5 were compared with our previous results. The performance results presented were consistent with our earlier results. These results represent RTI performance in the JADS test bed. Other architectures will likely produce different results. Federation developers need to test in their own environment to understand how a particular RTI will perform.

Table 2. RTI Best Effort Performance in One-Way Tests

Test Type	Losses Began	Max Latency during Runs	
		With Losses	With No Losses
IP Multicast	301 bytes at 500 Hz	147 - 452 ms	7 - 14 ms
RTI 1.3r4	301 bytes at 400 Hz	214 - 512 ms	9 - 17 ms
RTI 1.3r5	301 bytes at 400 Hz	214 - 512 ms	9 - 17 ms
RTI 1.3r6	301 bytes at 400 Hz	511 - 518 ms	14 - 28 ms
RTI 1.3NG beta	17 bytes at 100 Hz	3.3 - 7.2 sec	9 - 22 ms
Mak RTI 1.3.1b	301 bytes at 500 Hz	160 - 460 ms	8 - 17 ms

Table 3 summarizes the performance of TCP and RTI reliable transmission during the one-way tests. Instrumentation points were the same as in the previous test. The "break point" was where the receiving federate started experiencing problems, displayed RTI TCP error messages, and was no longer receiving data. The "good" runs excluded runs when the RTI broke.

Table 3. RTI Reliable Performance in One-Way Tests

Test Type	Break Point	Latency in Good Runs
TCP	301 bytes at 500 Hz	6 - 13 ms
RTI 1.3r4	301 bytes at 400 Hz	9 - 17 ms
RTI 1.3r5	301 bytes at 400 Hz	9 - 17 ms
RTI 1.3r6	301 bytes at 400 Hz	14 - 19 ms
RTI 1.3 NG beta	*	9 - 11 ms
Mak RTI 1.3.1b	17 bytes at 100 Hz	539 - 605 ms

* Never broke but maximum data publish rate achieved was 234 Hz

Since the worst case total load estimated for the JADS EW Test was approximately 100 bytes at 200 Hz, the performance of RTI 1.3r4 (the latest version available at the time of Phase 2) was considered acceptable for further screening (using the multiple federate test described in Section 5) prior to the JADS EW Test Phase 2. Phase 2 execution was hampered by a data loss problem and by federate "core dumps." RTI 1.3r5 was available for use in Phase 3 execution, so JADS took time to test it. JADS determined that RTI 1.3r5 fixed some of the problems with RTI 1.3r4 and did not introduce any new problems that would adversely affect the JADS tests. Therefore RTI 1.3r5 was used for Phase 3.

Table 4 shows the maximum latency observed in the one-way reliable tests with RTI 1.3r5. Unlike the previous data, these results were not obtained in the test bed but on the actual installed network. These results provide a glimpse into the differences between local test bench results and those obtained on an actual WAN. As indicated by the data in this table (e.g., 101 bytes at 100 Hz and 101 bytes at 200 Hz), there are other factors besides size and transmit rate that influence the latency. The items without data are where the RTI broke.

Table 4. RTI 1.3r5 Reliable Maximum Latency

	Packet Size					
Rate	**17**	**51**	**101**	**301**	**501**	**1001**
5	0.039	0.011	0.011	0.012	0.034	0.018
10	0.011	0.010	0.016	0.027	0.014	0.023
20	0.026	0.011	0.011	0.088	0.018	0.018
50	0.085	0.028	0.016	0.014	0.015	0.033
100	0.034	0.016	0.150	0.020	0.082	0.123
200	0.021	0.043	0.020	0.042	0.046	
400	0.019	0.022	0.127			
500	0.128	0.216	0.046			

5.4 Multiple Federate Test

5.4.1 Description

Once the one-way results were in the reasonable region, the next test used a multiple node, multiple federate test that was more representative of the Phase 2 federation. JADS simplified the Phase 2 federation to three federates with each federate representing a single facility. To simulate the AFEWES federate, we configured *testfed* on one computer to publish 10 attribute updates every 50 ms. To simulate the four federates at JADS, we configured another instance of *testfed* to publish 2 attribute updates every 50 ms. The ACETEF node was simulated by configuring *testfed* to publish 1 attribute update every 50 ms. All three federates published interactions at approximately 2 Hz. The size of attributes and interactions was 121 bytes. Attributes were published best effort. Interactions were published reliable. The test duration was five minutes. Fifteen runs were made in this configuration.

5.4.2 Multiple Federate Test Results

The results of the multiple federate tests with RTI 1.3r5 are shown in Table 5. The JADS federate was the only one that experienced attribute and interaction losses. Upon further examination of the data, it was discovered that all the losses occurred at the start of the run. The ACETEF and AFEWES federates were started first. The JADS federate was configured as the controller federate. This means it sent the start interaction to begin the test. While the ACETEF and AFEWES federates were waiting for the start they published dummy data at 1 Hz. There was no wait interval specified for the JADS federate. So as soon as it was started, it began publishing at 20 Hz. The actual start-up sequence used in Phase 2 and Phase 3 avoided this problem. While we had seen this problem in integration testing, these data illustrate the impact of publishing data immediately after start-up at a rate faster than 1 Hz. JADS documented problems with data loss at the start of runs when federates begin publishing immediately.

Table 5. Multiple Federate Test Results

Federate Name	ACETEF	AFEWES	JADS
Attributes Published (per sec)	20	200	40
Interactions Published (per sec)	2	2	2
Attributes Received (per second)	240	60	220
Attributes Lost	0	0	22 - 44
Attribute Maximum Latency	127	114	100
Attribute Mean Latency	20	17	20
Interaction Maximum Latency	97	41	38
Interaction Mean Latency	22	19	22

6.0 RTI Performance During the JADS EW Test

Table 6 shows performance data of the RTI during the JADS EW Test. Federate problems that caused a run to be aborted included the crash of one or more federates and lost TSPI data from the platform federate. Procedure problems were usually due to the loading of an incorrect script. The yield of usable runs was higher for Phase 3 (88% versus 72%) indicating that RTI 1.3r5 was more stable than 1.3r4.

As mentioned earlier, the acceptable end-to-end latency (from radar mode change to jammer response) was determined to be 500 ms. Any runs with an end-to-end latency greater than 500 ms were considered bad. The Phase 3 tests (using RTI 1.3r5) had only one bad run (589 ms for the end-to-end latency). The maximum latency for the good runs was 10% higher for Phase 3 (417 ms versus 380 ms). However, the mean value for Phase 3 was 35% lower than Phase 2 (167 ms versus 255 ms) once again indicating that the later release had better performance.

Table 6. EW Test RTI Performance

	Phase 2	Phase 3
RTI Version	1.3r4	1.3r5
Total runs (excluding excursions)	341	255
Runs aborted due to federate problems	83	20
due to procedure problems	10	10
due to network problems	2	1
Usable runs (total - aborted)	246	224
Runs with unacceptable latency (bad runs)	8	1
Good runs (usable - bad)	238	223
Maximum latency in bad runs (ms)	12352	589
Latency in good runs (ms)		
minimum	114	113
mean	255	167
maximum	380	417

7.0 Technical Lessons Learned

7.1 No Plug-and-Play for High Performance Federations

RTI 1.3r4 (used in Phase 2) and RTI 1.3r5 (used in Phase 3) were adequate for the JADS tests. However, we determined the JADS EW Test performance requirements early in the test development process. We tested many versions of the RTI within the context of these requirements. We started RTI testing well in advance of our actual tests (about one year). We found problems with the RTI, but we had plenty of time to have the problems fixed or experiment and develop a workaround for them. If you have specific performance requirements for your federation, you may be able to achieve them. However, it will take some work. Before your performance requirements are met, plan on spending some time experimenting with runtime infrastructure initialization data (RID) parameters, your federate implementation, tick, and other options in your architecture.

7.2 Data Packet Size

The largest JADS message (attribute or interaction) was 94 bytes. However, by the time the message reached the network, the size of the message was 298 bytes. This additional overhead came from the Ethernet header and checksum, the IP header, the UDP header, and the RTI. The RTI contribution to this overhead was approximately 70 bytes.

7.3 Network Architecture

The JADS EW Test network was initially configured using an unswitched, half duplex, 10BaseT LAN. JADS RTI tests occasionally had reliable latency values of more than one second. The cause of these high latency spikes has been traced to excessive Ethernet collisions. A switched, full duplex, 100BaseTX LAN network was installed and the number of Ethernet collisions dropped to zero. The one-second latency spikes were eliminated. The JADS EW Test might not have been successful if it had been executed using the unswitched, half-duplex configuration of the network.

7.4 TCP Implementation

RTI reliable traffic is published using transmission control protocol (TCP). TCP implementations vary from operating system to operating system. The algorithms used to retransmit corrupted or lost packets (e.g., due to Ethernet collisions) can add significant latency (more than one second). The JADS EW Test was able to avoid this problem by configuring the network as a switched (versus a shared) Ethernet. Some TCP implementations (e.g., IRIX 6.3) use the Nagle algorithm to reduce network traffic. This algorithm causes the TCP implementation to hold a message for up to 200 ms in the event that an acknowledge or other message can be sent in the same network packet. To minimize latency, a federation with additional TCP connections (other than those created by the RTI) may want to disable the Nagle

algorithm by setting the TCP_NODELAY socket option. The RTI disables the Nagle algorithm, but it provides options in the RTI.rid file to perform bundling if desired.

7.5 Single Processor Computers

Single processor computers are more difficult to use in high performance federations. The single processor is a resource that has to support the operating system, the simulation software, the RTI local component, and any other software (e.g., the JADS logger) or hardware (e.g., a time card). Two factors in making all this work are the arguments supplied to the RTI tick service that allocates processor time to the RTI and the size of the operating systems time slice that is the minimum amount of time the operating system will allocate to an executing process. JADS found that, in some cases, the size of the operating system's process scheduling time slice could affect latency. However, this only affected federates executing on single processor computers. The default time slice on SGI computers (defined by the kernel parameter slice_size) was 30 ms. Table 7 shows the results of a simple one-way test between two computers on the same LAN using the zero-argument form of the RTI tick service comparing time slices of 10 ms and 30 ms.

Table 7. Time Slice Comparison

Time Slice Size	30 ms	10 ms
Minimum Latency	14 ms	6 ms
Maximum Latency	110 ms	56 ms
Mean Latency	34 ms	25 ms
Max Time in Tick (sender)	113 ms	13 ms
Max Time in Tick (receiver)	98 ms	42 ms

When a federate is executed on a single processor machine, other software running on that machine can increase latency of data to and from the federate. Graphics updates in particular can severely impact latency. The JADS team noticed that a graphical screen saver caused latency spikes in the hundreds of milliseconds. High latency spikes were also caused by network file transfers and remote logins.

7.6 Process Sleep

Since the federate and the RTI are single threaded, care must be taken when using sleep (or other variant such as sginap) within the federate. If the federate sleeps, the local RTI component (LRC) is also sleeping. While a federate (and its LRC) sleeps, it is possible for the entire federation to be waiting.

7.7 Data Collection

The HLA provides a flexible environment for linking simulations for distributed tests. Federations are free to define the format of messages exchanged among federates via the FOM and SOM. However, this flexibility complicates data collection. Since there was not a protocol

for data exchange (as in DIS protocol data units), a stealth logger that recognized and recorded all simulation traffic in a log file could no longer be connected to the network.

The JADS EW Test team chose to implement an RTI interface logger for data collection. An interface logger resided between the federate software and the application program interface (API) to the RTI (Figure 2).

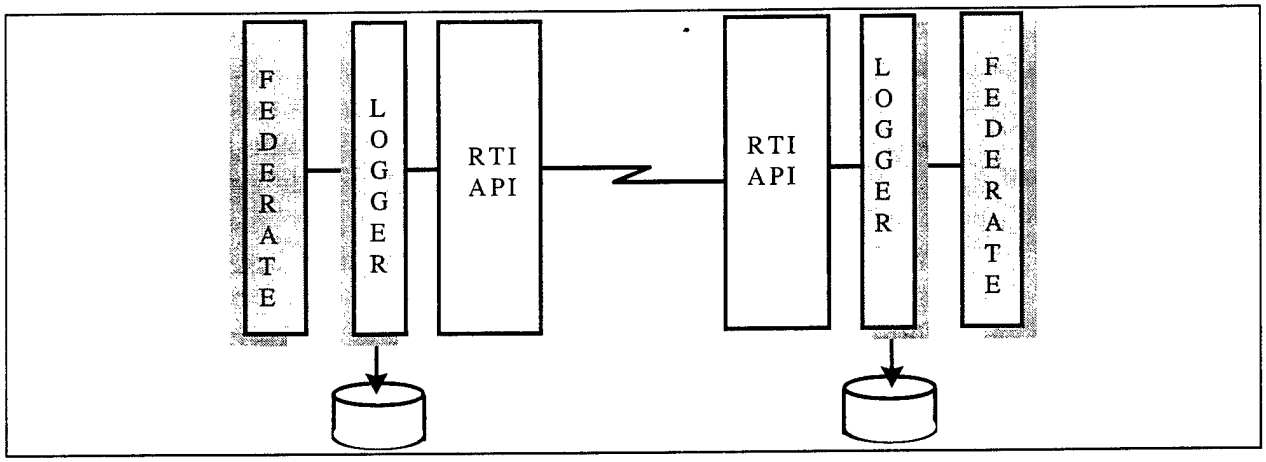


Figure 2. RTI Interface Logger

An interface logger was chosen to accurately determine federate to federate message latency. Using an interface logger, messages were time stamped as they were sent to the RTI at one federate and when they were received from the RTI at another federate. Every message header also contained the data creation time.

8.0 Runtime Infrastructure

8.1 Tick

The RTI software is single threaded. A federate must call the tick method to yield time to the RTI. The tick method exists in two forms – one taking zero arguments [tick()] and another taking two arguments [tick(minimum, maximum)]. The zero-argument version yields time to each major activity within the LRC. A typical activity would be draining inbound event queues and providing callbacks to the federate via the FederateAmbassador. The two-argument version of tick also yields time to the LRC but suggests lower and upper bounds on the time being allotted to tick. If the specified minimum time interval has not elapsed after all available processing has been done, the LRC will pause until the minimum time interval has elapsed, then return immediately.

In the zero-argument form of tick, the LRC completes all the work in its queues before it returns to the federate. This can potentially starve the federate of central processing unit (CPU) cycles. Using the two-argument form, the LRC will attempt to empty its work queues. However, if the maximum time is reached before the work is completed, the LRC will return to the federate. If

the LRC completes its work before the minimum time has elapsed, it will attempt to use up the remainder of the time before returning to the federate thread. It does this by blocking the process. The LRC becomes unblocked and returns to the federate thread after the minimum time has expired. The LRC will process any incoming messages while the process is blocked.

In JADS RTI performance tests, it was noticed that the zero-argument form would occasionally have high latency spikes. Latency results comparing the two forms of tick in a simple one-way test are shown in Table 8. For this test, one federate published only reliable attributes and interactions and the other federate only subscribed. The default slice size of 30 ms was used for these tests. The latency differences were due to the fact that the federates were executed on single processor machines. The zero-argument form never blocks the federate/RTI process. When software executes on a computer with only one processor it must contend with the operating system. Unless the federate performs input/output or blocks intentionally using some other method (like sleep), the zero-argument form of tick will not cause the federate and the LRC to block. The operating system will only wait so long for an opportunity to complete its work and then it will take control of the processor. The federate will have to wait to regain control of the processor. This manifests itself as additional latency.

Table 8. Tick Comparison

	tick()	**tick(.010,.020)**
Minimum Latency	14 ms	6 ms
Maximum Latency	110 ms	57 ms
Mean Latency	34 ms	30 ms
Max Time in Tick (sender)	113 ms	75 ms
Max Time in Tick (receiver)	98 ms	25 ms

Since most federates in the JADS EW Test federation executed on single processor computers, the two-argument form of tick was used with a minimum value of 10 ms and a maximum value of 20 ms. This produced better latency results for the data published by the JADS federates. It should be noted that occasionally the RTI would use more than the maximum time specified.

As federation developer, you must experiment with tick, number of federates and typical publication rates. If latency is not an issue or you are executing your federates on multiple processor computers and the LRC is not starving your federate, you may be able to use the zero-argument form of tick. This will relieve the federate of the responsibility of determining how much time to give the LRC. If the federate executes on a single processor computer or the LRC must deal with a lot of federation activity, you should probably use the two-argument form of tick.

8.2 Attribute/Interaction Structures

The implementation of attributes and interactions can affect performance. You can define each piece of data that can be associated with an object as a separate attribute as in the following excerpt from a FED file.

```
(class Live_Entity_State
        (attribute Data_Header best_effort receive)
        (attribute Object_Type best_effort receive)
        (attribute Object_Velocity best_effort receive)
        (attribute Object_Location best_effort receive)
        (attribute Object_Orientation best_effort receive)
        (attribute Object_Acceleration best_effort receive)
```

Or you can define a structure that contains all the data and only declare one attribute represented by the structure as in the following.

```
(class Aircraft
        (attribute Live_Entity_State best_effort receive))
```

```
Construct Live_Entity_State
    Data_Header                 header
    Object_Type                 type
    Object_Velocity             velocity
    Object_Location             location
    Object_Orientation          orientation
    Object_Linear_Acceleration  acceleration
```

We ran a series of tests comparing the performance of the two methods of declaring attributes. In one case we declared twenty individual attributes for a test object (similar to the definition of an entity state object in the real-time platform reference [RPR] FOM) with a total size of 168 bytes. In another case we declared one attribute for a test object with a size of 168 bytes. We ran simple one-way tests with each object. There were no performance differences until we started publishing at 400 Hz. The tests with the individual attributes started experiencing data losses and high latency in the best effort tests and caused an RTI internal error in the reliable tests when the data were published at 400 Hz. The tests using the attribute structure did not experience these problems until data were published at 600 Hz.

8.3 Reliable Data and Network Bandwidth

By default every federate is configured with its own reliable distributor (RELDISTR). When a federate publishes a reliable message, its RELDISTR transmits the message to every RELDISTR in the federation. It's important to note that unless the federation uses data distribution management (DDM), all reliable messages will be sent to every RELDISTR whether the federates connected to that RELDISTR have subscribed to the data or not. The subscription filtering is performed by the LRC upon receipt of the message. So even if a federation minimizes which federates subscribe to data in an effort to reduce network traffic, the data will be sent to all federates if DDM is not used.

One way to minimize network traffic is to configure a LAN with a single RELDISTR. All federates on the LAN use the same RELDISTR to send reliable data to other federates at remote locations. In the JADS EW Test federation, there were five federates in the TCAC in Albuquerque. These federates were configured to use the RELDISTR created by the runtime infrastructure executive (RTIexec). Further information on how to configure RELDISTRs in a federation can be found in the "Reliable Service in RTI 1.3" paper found within the RTI.rid file editor help pages.

8.4 Multicast Groups

In the RTI 1.3 series developed by DMSO, as federates join a federation, multicast groups are created to handle the transmission of best effort data among the federates. A "broadcast" multicast group is also created to handle best effort traffic sent to all federates. Using default settings in the RTI.rid file, the RTI only initializes enough multicast groups to accommodate the best effort traffic among three federates. Once a fourth federate joins a federation, all best effort data destined for that federate (or any subsequent federates) is transmitted using the "broadcast" multicast group. The consequence of this is that all best effort data sent to the fourth (and subsequent) federate are sent to all federates regardless of whether the federate subscribed to the data or not. The LRC filters data that its federate has not subscribed to. Another RTI may or may not work the same way. Federation designers need to understand how the RTI.rid file settings affect the messages that the network and federates have to handle.

8.5 RTI.rid File Parameters

The RTI.rid file contains user-modifiable parameters that allow the developer to optimize the federation execution to achieve desired results (e.g., lower latency versus high throughput). As stated above, the JADS EW Test federation used a single RELDISTR for the federates in the TCAC. This was achieved by modifying the parameters auto_reldistr_config, reldistr_on, auto_discover_on, and discov_string.

The JADS development team wanted to minimize latency on all messages in the federation. A feature of the RTI that increases throughput (and can also increase latency) is bundling. If bundling is turned on, the RTI will hold on to messages for a period of time so that multiple messages can be sent in a single data packet. To minimize latency, bundling must be turned off. The RTI.rid file parameters tcp_bundling_toggle and udp_bundling_toggle control bundling. Prior to RTI 1.3r5 bundling was turned on by default. Starting with RTI 1.3r5 bundling was off by default.

TCP and UDP polling_interval were the only other parameters modified by the JADS EW Test federation. These identified the minimum amount of time between polls to check for incoming network traffic. The default value for these parameters was 5 ms. The JADS EW Test federation used a value of 0. A smaller interval means the network socket will be polled more frequently. This reduces the likelihood of packets being dropped because of a filled socket buffer. Larger intervals conserve CPU cycles.

8.6 Federate Join, Publish, and Resign

During RTI tests if, after a federate joined, it immediately began publishing data at its normal data rates (e.g., 20 Hz), some best effort data were lost and reliable data experienced high latency. If a delay of a few seconds was added at the start when the federate published data at a low rate (e.g., 1 Hz), then the initial losses and high latencies did not occur.

It was also noticed during the JADS EW Test Phase 3 that high latency on reliable data and dropouts on best effort data occurred when a federate joined late. Data recorded by EtherPeek (a network packet analysis tool) showed that the RTI transmitted a large number of messages, some of which were quite long when compared to the longest JADS message, when the late federate joined. To minimize latency on the first messages published, the JADS team recommends that, if possible, publication should be synchronized to begin only after all federates have joined.

It is important that federate shutdown be synchronized as well. After a federate resigns from a federation, it must continue to tick the RTI for a few seconds so that the resignation is handled gracefully.

9.0 Anomalies from Previous RTI Versions

The problems documented in this section were identified in previous versions of the RTI and were fixed in subsequent versions (no later than RTI 1.3r6). Normally these would not be included in a report of this type; however, we found some of these problems when we briefly tested both RTI 1.3 NG (beta) and the Mak RTI indicating that there may be little if any communication among RTI development teams. If you use an RTI from another vendor and it exhibits these symptoms, it may be related to these problems.

9.1 Reliable Message Buffering

In some of our early RTI tests, we noticed that messages were being buffered when publish rates exceeded 5 Hz (see Figure 3). Upon further investigation, we determined that the buffering was caused by the IRIX TCP implementation of the Nagle algorithm. The Nagle algorithm buffers small packets on the transmit side for a period of time in the event other messages are being sent to the same node and thus can be sent in the same network packet. On SGI computers, the network can wait up to 200 ms before sending the buffered packets.

The Nagle algorithm can be controlled using the TCP_NODELAY socket option. If TCP_NODELAY is set to TRUE, then the Nagle algorithm is turned off. On the SGI computers, the default value for this option is FALSE. Prior to version 1.3r2, the RTI ran with the operating system default setting for the TCP_NODELAY socket option. This means that the Nagle algorithm was in effect for both attribute and interaction data sent reliable. If data are published using reliable transport mode at data rates at or above 5 Hz, then the latency of the data increases. As a result of sharing this information with RTI developers, RTI version 1.3r2 sets the TCP_NODELAY option to TRUE, disabling the Nagle algorithm. However, there are RTI.rid file parameters that can be used to buffer messages.

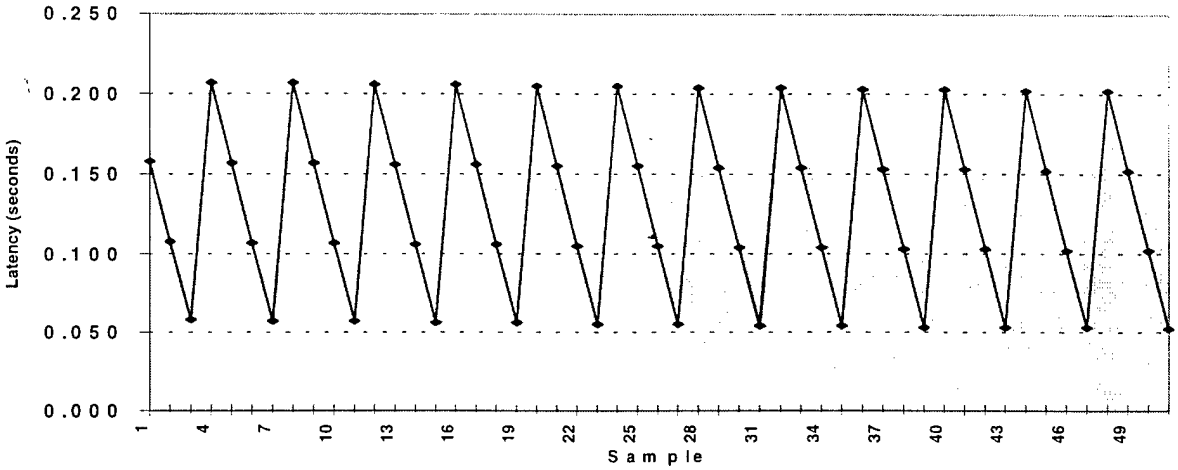


Figure 3. Latency 101 Bytes at 20 Hz

9.2 Multicast Time To Live

In initial tests performed with RTI 1.0r2, best effort traffic was not received at any computer on a different LAN. Using the network packet "sniffer" tool to look at the network data packets, a JADS network engineer discovered that the time to live (TTL) value was set to one. A packet's TTL indicates how many hops it can take before it is discarded by the network. A value of one does not allow a packet to exit the LAN, i.e., to pass through a router to reach a system on another LAN or a WAN. Hence, a federation running with RTI 1.0r2 out of the box would not allow federates to communicate best effort traffic outside of a LAN. Using the JADS 2-node network configuration (shown in Figure 4) required network data packets to cross from one LAN through the routers (Micro-IDNX-20) to reach the test federate on another LAN mirroring the EW Test Phase 2 network architecture. DMSO provided a special library that allowed JADS to use RTI 1.0r2 across our network communications gear. Subsequent versions of the RTI provided for a user-defined parameter value in the RTI.rid file to set the TTL. This problem was observed in the beta version of RTI 1.3NG and the MAK RTI.

9.3 Excessive Best Effort Data Loss

Prior to RTI 1.3r5, certain conditions in a federation could cause loss of best effort data from one or more federates. The loss could last for many seconds and recover. Or it could be permanent and never recover. Leading up to the Phase 2 tests, JADS experienced this problem intermittently with the platform federate (publishing aircraft TSPI at 20 Hz). There were times when some federates would not receive the TSPI data at the start of a run. It turned out that the federate join sequence affected the problem. JADS was able to institute a join sequence that kept the occurrence of the problem to a minimum. In general, to reduce this problem, the federate that joins first should not publish any high-rate, best effort data if possible.

10.0 Unexplained Anomalies

10.1 Best Effort High Latency

You usually do not see abnormally high latency on data sent best effort. If a problem is encountered in the multicast transmission of a message, the message is simply dropped. This is not necessarily the case in architectures with the RTI installed. Phase 3 run 2 had higher than normal latencies on some best effort data received by the AFEWES federate. The maximum latency observed was more than one second. Normal latency for these messages should be less than 100 ms. The expected behavior of the network itself is to drop UDP multicast messages older than one second.

10.2 Latency Spikes

Figure 4 shows a latency spike that occurred in the one-way reliable tests using RTI 1.3r5. This spike occurred while publishing 101 bytes at 400 Hz. Similar spikes occurred at all data rates with all versions of the RTI as well as the raw TCP tests (without the RTI). The operating system and the fact that the tests were executed on single processor computers may have caused the problem. Latency spikes might be eliminated if the federates are executed on multiple processor computers.

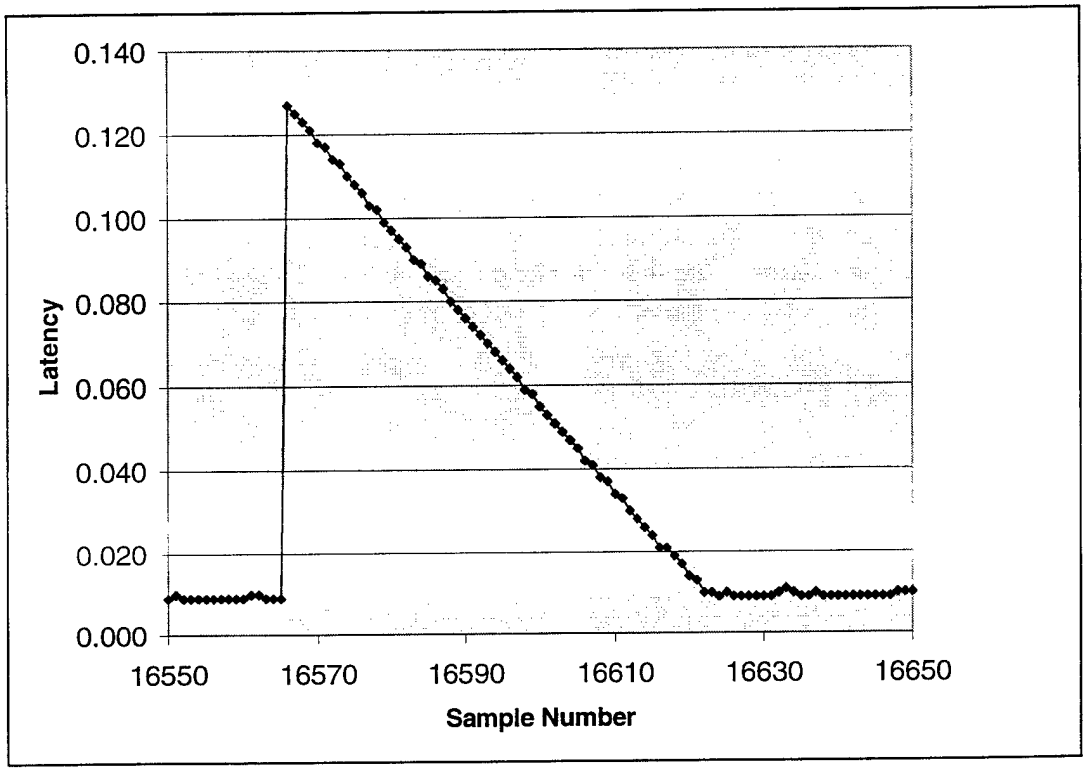


Figure 4. RTI 1.3r5 Reliable Latency Spike Publishing 101 Bytes at 400 Hz

10.3 Differential Latency

At times during the JADS EW Test, a message sent to multiple federates (using reliable transport) had significantly different latency at each of the receiving federates. In some cases the receiving federates were at different sites. There were also cases of the receiving federates residing on the same LAN (in the TCAC in Albuquerque) having significant differences in latency.

Phase 3 run 28 is an example of the differential latency at different sites. ACETEF sent a burst of system under test (SUT) receiver track update (RTU) messages followed by two SUT jammer technique command (JTC) messages. In either the AFEWES log file summary or the JADS log file summaries, there was an abnormally high latency (> 200 ms in many cases) for the second JTC, but normal latency for the first. Also, if there was a high latency for the second JTC message in the AFEWES summary, then there was a normal latency in the JADS summaries, and vice versa. Perhaps, this indicates some kind of transient response problem in the RELDISTR. Or, it might be a transient response problem in the IRIX TCP.

Phase 3 run 121 had examples of differential latency on the TCAC LAN. A JTC message was sent from the ACETEF federate. Most of the federates in the TCAC received it with, at most, 88 ms of latency. The latency on this message to the TCF federate (also in the TCAC) was 225 ms.

10.4 Reliable Data Loss

By definition, reliable data are not supposed to ever get lost. They may arrive very late, but they should never be lost. During run 146 of Phase 3, a reliable message was lost. It appeared in the publishing federate's log file. The network sniffer data on the sending side confirm the message was transmitted. The network sniffer data on the receiving side show that the message was received and a TCP acknowledgment was sent back to the publishing federate. This implies that the receiving federate's LRC received the message. However, the message never appeared in the log files for any of the federates that were supposed to receive it. So the message was never passed from the RTI to the federate. Looking at the network analysis tools, there were no indications of any problems or errors, or excessive traffic at the time that the message was sent. It seems as though all six of the receiving LRCs lost the message.

11.0 HLA Application to Other Types of T&E

In this section we look at how ADS applied to the other two JADS tests. The section concludes with our assessment of the utility of HLA to T&E in general.

11.1 HLA Application to the End-to-End Test and Legacy Simulations

JADS found the strongest utility for distributed testing in the area of C4ISR systems. The JADS End-To-End (ETE) Test was designed to evaluate the utility of ADS to support testing of C4ISR systems. The test focused on the Joint Surveillance Target Attack Radar System (Joint STARS)

as one component of a representative C4ISR system. The ETE Test also evaluated the capability of the JADS TCAC to control a distributed test of this type and remotely monitor and analyze test results.

The ETE Test consisted of four phases. Phase 1 developed or modified the components needed to develop the ADS test environment. Phase 2 used the ADS test environment to evaluate the utility of ADS to support DT&E and early OT&E of a C4ISR system in a laboratory environment. Phase 3 transitioned portions of the architecture to the E-8C aircraft, ensured that the components functioned properly, and checked that the synthetic environment interacted properly with the aircraft and actual light ground station module (LGSM). Phase 4 evaluated the ability to perform testing and evaluation in an ADS-enhanced live test environment.

In 1994, the JADS ETE Test began the development of a radar simulation, the Virtual Surveillance Target Attack Radar System (VSTARS), that represented the Joint STARS E-8C aircraft and communicated with the Army's ground station module (GSM). This simulation would receive target data by interfacing with a DIS network that utilized IEEE Standard 1278. JADS found that for one phase of the test, the data provided in the entity state protocol data units (ESPDU) had to be reduced to fit within the constraints of a satellite communications (SATCOM) link to the E-8C aircraft. To solve this problem, JADS developed the concept of ground and air network interface units (NIU). This concept had a ground NIU (GNIU) that received and sent DIS protocol data units (PDU) - primarily ESPDUs. The GNIU then processed the ESPDU - dropping all the fields that were not used by VSTARS, performing coordinate conversions between World Geographic System 84 and Topocentric Coordinate System, and reducing the accuracy of the coordinate attributes from 32 bits to 16 bits before passing it to the air NIU (ANIU), either via SATCOM or system bus. In an HLA environment, the data requirements for VSTARS could be limited to the reduced data sent to the ANIU, thus simplifying the interface requirements. However, JADS decided to proceed with the use of DIS for the test utilizing VSTARS because we decided HLA was too immature to use in the effort. (The contracts for VSTARS were awarded prior to the HLA protofederation efforts.) Continued use of VSTARS mandates that it become HLA compliant prior to the first day of fiscal year 2001 or be retired.

JADS worked with the VSTARS developers to identify an executable DIS to HLA migration strategy. JADS identified several issues that needed to be addressed in the migration strategy. The key issue for the RTI was the number of objects that Joint STARS was capable of tracking and that VSTARS would be able to handle. Although the JADS test presented the VSTARS with more than 9,000 individual target objects during our tests, more recent uses of VSTARS have had requirements approaching 100,000 targets. We were curious about how many objects the RTI was capable of supporting, so we ran a test to see how many objects could be declared before the RTI failed. The results indicated that the current RTIs might have to be altered to support federations that require 100,000 objects to be visible to a single sensor.

The second issue that affected the migration strategy came from the platform hardware and operating system that VSTARS uses. Since VSTARS is essentially a set of software interfaces and models that allow the actual Joint STARS operational flight programs (OFP) to interact with

a synthetic environment, the VSTARS is hosted on the same hardware (DEC Alpha) and operating system (Open VMS) as the OFP. That combination was not supported by any of the available RTI versions. Since it was impractical to migrate VSTARS, and we were concerned about the performance of an RTI designed and tested in a SUN/SOLARIS environment being compiled for ALPHA/Open VMS, we recommended using a gateway that would host the RTI and format the data messages for VSTARS.

The final issue stemmed from the potential uses of VSTARS. The projected uses encompass the spectrum from a simulation tool to evaluate and perform trade-offs on new technology, a stimulator for test and evaluation, a stimulator for mission crew training, a mission rehearsal tool, and a simulation of Joint STARS for exercise support. Each of these applications has a different set of requirements, interfaces, and customers. At the same time, resources and configuration management issues dictate a minimum number of VSTARS configurations and interface strategies. They are faced with the development of a flexible interface capable of supporting a variety of requirements and capable of operating with a variety of RTIs. Such an interface would be a challenge for a team with a lot of experience in distributed simulation, much more so to a team of radar designers with a minimum of simulation and distributed simulation experience. From the standpoint of legacy simulations, especially those designed for test and evaluation, we believe the need to develop a flexible interface to a single simulation or test facility will be a common requirement. Future developers of such interfaces will need to carefully analyze their requirements and options to be successful.

11.1.1 RTI Object Declaration Tests

In the RTI object declaration tests, JADS attempted to register a certain number of objects, publish one update for each object, and then resign. The updates were published at 100 Hz. Two federates were used in the test. One registered the objects and sent the updates and the other only subscribed to the attribute and received the update.

Table 9. Object Declaration Test Results

RTI	Max Objects	Time to Register	Time to Resign	Data Rate Attempted	Data Rate Achieved
RTI 1.3r6	9500	315 sec	330 sec	100 Hz	100 Hz
RTI 1.3 NG beta	10,000	45 sec	41 sec	100 Hz	26 Hz
Mak RTI 1.3b	10,000	128 sec	0	100 Hz	78 Hz

JADS found that if you do the test with only one federate, you can register more objects without a problem. One-federate tests may be the basis for some claims about how many objects some RTIs can register. However, we're not sure how meaningful it is to be able to register objects and publish attributes if no other federate is around to receive the data.

For RTI 1.3r6

- Both federates must join the federation before the objects are registered. If the receiver waits until all objects have been registered before it joins, it will get an RTI internal error.
- Tick the RTI for 1.5 seconds during the register process every 100 objects. If you don't do this, the sender gets an RTI internal error.

For RTI 1.3 NG beta

- The sender registered all objects before the receiver joined. When the receiver joined before the registration started, it took 20 minutes to register and discover the objects.

For Mak RTI 1.3b

- Since the other RTIs had a limit of approximately 10,000 objects, Figure 9 shows the performance of the Mak RTI with 10,000 objects. However, the Mak RTI was able to register and update more objects. It took fifteen minutes to register 25,000 objects. But only a 29 Hz update rate could be achieved while trying to publish at 100 Hz. The RTI was able to register 75,000 objects in 80 minutes. But the publication of the data at 100 Hz, which should have lasted 12.5 minutes, took more than an hour before the program was terminated.
- The resign of both federates was immediate.

11.2 HLA Application to the System Integration Test

JADS found utility for testing precision guided munitions using distributed testing techniques. SIT was executed in two different DIS-based architectures. In preparing this report, JADS re-examined the SIT to determine what, if any, concerns would have been introduced by replacing DIS with HLA. Two concerns were identified. The first was latency. While neither test was a true closed loop between the shooter and the target (the target did not respond to the missile firing), the one-way latency achieved in the Linked Simulator Phase was 100 ms from data production to consumption. This latency should be achievable in HLA-based architectures. However, the data would have to be passed using best effort and multiprocessor computers might have been required as well. It is possible that data loss might have been an issue for the one-time messages such as start and stop, but it is not likely to have been much worse than the DIS broadcast versions of the same messages. Data loss was not an issue for SIT. JADS believes that an HLA-based architecture could be developed using a current RTI that would meet SIT latency, although the use of multiprocessor computers would increase the cost of the architecture. HLA addresses this concern.

The second concern was the use of non-HLA links in the architectures. SIT used two non-DIS links, one in each phase. The Linked Simulator Phase required a 1553 data link from the cockpit simulator acting as the shooter to the missile seeker laboratory. The data link was required to pass the initialization messages to the missile. Signal timing constraints prevented this link from working in DIS. The solution was to use a native 1553 link between the facilities. The Live Fly Phase required telemetry links from the live aircraft on the range. These signals were converted to DIS messages by the ground station receiving the signals. These are both cases where a non-HLA-link would be required to supply critical information into the federation. Since these would

not be documented in the FOM according to HLA Rule 3, it points to the need for an interface control document (ICD) that would cover more than the FOM. These types of links are likely to be very common in the T&E environment. The current HLA allows the federation designer to use additional documents and tools in federation development. HLA addresses this concern.

11.3 General Utility of HLA to T&E

JADS demonstrated that HLA can support T&E federations. As T&E makes use of distributed testing technology, the community will use the protocols and connecting hardware that make sense. For example, telemetry is likely to be used where live aircraft on an open air range are involved in a distributed test. This means that the OMDT and the FEPW are not the only tools a test designer will need to create distributed tests when HLA is only one of the linking architectures. This will also stress the definition of "HLA compliant."

HLA promises to provide access to simulations that are otherwise unavailable to the tester. This will allow the tester to develop richer synthetic environments for T&E. This also places a demand on the test designer, the simulation owner, and the RTI developer. The test designer will have to take the lead in developing the FOM and other ICDs needed to bring together a successful federation. Simulation owners will have to implement flexible interfaces to allow the simulations to be used in a variety of federations. RTI developers need to work with test developers to create RTIs with the right performance characteristics to be useful to the tester.

As the VSTARS experience demonstrates, there may be barriers to migrating non-HLA simulations to HLA that each simulation owner will have to address. As with all models, HLA simulations have long-term ownership issues and costs that federate and federation developers need to understand.

As the SIT experience demonstrates, there is a real requirement for an ICD that encompasses the entire effort. The need for native protocol communications among players to occur outside of the RTI is real. T&E federation designers will have to understand how to make non-HLA players interact with HLA federations in a common exercise.

12.0 T&E HLA Requirements

JADS found that, in general, there were no unique T&E requirements that were not being met by the HLA. There are some areas of concern that the T&E community must understand to use HLA effectively. The concepts of re-use and the implication of the HLA to the long-term ownership cost for any simulation are not well understood at this time. HLA has several features that make it attractive to simulation developers and owners of legacy simulations that impact the ability of other federations to use the simulation. These features are HLA compliance, the lack of required standard data messages, and RTI interoperability.

Compliance testing is required to meet the intent of Dr. Kaminski's memorandum directing the use of HLA by all simulations. The HLA compliance test requires that the simulation interact with a federation but does not specify what federation. Those looking to include a particular

simulation in a federation will have to look past the compliance certification to understand more about how the compliance test was done (what federation was used in the test, what RTI version was used, what RTI services were used, etc.) and how well the simulation itself functions (has the simulation been used for the intended purpose, are there limitations to the simulation, has the simulation been through verification/validation, etc.). HLA compliance alone is not sufficient reason to include a particular simulation in your federation.

HLA provides flexibility to existing simulations and to the T&E community because there are no required standard data formats or data messages that the simulation must implement. However, this flexibility comes with a cost. Each federation must, through whatever process, reach agreement on the data messages and data dictionary that will be used by the federation. There is no guarantee that the data messages used in federation A will be the same as federation B even if they use most of the same federates. This means that simulations will have to be designed with some flexibility to allow them to interact with different federations or will be required to develop modifications for each federation. T&E resources are generally used by more than one customer. This means those resources will have to design flexibility into their interface. This also means that careful configuration control is necessary to ensure the right interface is installed for each given federation. T&E will undoubtedly take advantage of the flexibility, but designers need to understand the implications.

The final impact to reuse is RTI interoperability. RTIs are not currently interoperable, and many in the field do not believe this is a requirement. Requiring RTIs to implement services in the same way is likely to stifle commercial competition in RTI development. Performance breakthroughs in one vendor's RTI product could be lost when it has to interact with a competitor's product. However, since all RTIs are supposed to implement the same specification, the federate owner should be able to swap out RTI versions with little effort, assuming the desired RTI is available for the federate hardware/operating system combination. JADS found this to be generally true for the limited number of services used in the EW Test federation. The issue is not the ability to interchange RTI versions, but the real potential for differences in performance between RTI versions. Not only should each port of an RTI be optimized for the target operating system, but also each federate owner and federation will likely have to perform further optimization for their application. Additionally, each T&E federation will have to conduct a fairly rigorous integration effort, thus increasing cost of ownership. All the optimization and integration will be done as often as needed. If the federate owner is able to exactly recreate the configuration needed for each federation, then the optimization/integration is done once. Therefore, as each RTI is installed and optimized, the federate owner will become responsible for documenting the hardware and software configuration so that it can be restored for future executions with that federation.

While the above ownership issues are not unique to T&E, the issues need to be clearly understood by the T&E community. Where the issues impact performance, it is critical that the T&E community understand and address them as early in federation development as possible. Where the issues impact the ease of being used in multiple federations, the T&E community needs to understand the issue to reduce its cost of ownership as well as realize that the cost of creating a federation may increase if other simulations didn't design for reuse.

12.1 HLA Rules

JADS did not identify any additional T&E requirements for the HLA rules.

12.2 HLA Interface Specification

JADS did not identify any additions to the interface specification required by T&E.

12.3 RTI Services

The JADS federation did not use time management, ownership management, or DDM services because of the additional overhead incurred by these services. Time management services guarantee the time-ordered delivery of messages throughout the federation. Ownership management services allow federates to transfer ownership of objects to other federates. DDM services allow the federation implementers to partition the simulation into filtering regions and limit a federate's knowledge of the federation execution to a specific region. DDM services can reduce the network load because filtering is performed by the publishing federate. The default for the RTI 1.3 family is to have subscription filtering performed by the receiving federate.

Table 10. RTI Services Used by JADS Federates

createFederationExecution
destroyFederationExecution
discoverObjectInstance
getAttributeHandle
getInteractionClassHandle
getObjectClassHandle
getParameterHandle
joinFederationExecution
publishInteractionClass
publishObjectClass
receiveInteraction
reflectAttributeValues
registerObjectInstance
resignFederationExecution
sendInteraction
subscribeInteractionClass
subscribeObjectClassAttributes
updateAttributeValues

12.4 RTI Performance

T&E will likely choose RTI performance over RTI services. Performance has many dimensions. JADS identified two areas of interest to the T&E community. The first was latency. Federations that are examining the interaction of computer systems or the simulation of high-speed, highly maneuvering platforms are interested in reducing latency. JADS found that considerable performance improvements could be made using the current generation RTIs hosted on multiprocessor computers. Using best effort transmission exclusively also reduces latency and allows the federation designer to use less expensive LAN components (since multicast UDP is not affected by collisions.) JADS also found that the RTI induced a variation into the latency. Reducing the RTI component of latency variation provides the designer with more repeatable results.

The second dimension of performance was the size of the object space that any single federate can address. In our investigation of migrating a simulation of Joint STARS to HLA, we learned about the limits on the size of the object space that the current generation of RTIs can deal with. The Joint STARS was capable of detecting and imaging well over 10,000 objects. However, RTI 1.3 release 6 was capable of declaring only 9,500 objects. For testing the capacity of the RTI for objects, we had one federate declare objects, another federate subscribe to the attributes of the objects, and the first federate publish one update for each object. There were cases where more than 10,000 objects could be declared as long as no federate subscribed to them. However, this would not be a very useful federation.

As T&E gets more experience with HLA, more dimensions of performance will be identified. Some of these will be addressed as the communications protocols and object-oriented software technologies evolve and as RTI developers try to take advantage of other technology such as telemetry protocols. It is unrealistic to expect that a single RTI will ever be produced that will meet all the T&E performance requirements. Commercial development efforts are likely the key for the T&E community to get the RTI products they will need in the future.

12.5 HLA Support Tools

Most of the tools being created for HLA were too immature to be of much use during the development of the JADS EW Test federation development. As such, JADS did not evaluate them for T&E applications. However, JADS did find one requirement in the area of loggers. JADS developed its own logger to accurately measure latency between federates. The JADS logger was simple and it logged messages as they were moved into and out of the RTC. This style of logger was required to accurately measure latency, to identify bottlenecks in the architecture, and to unambiguously resolve temporal order of events. Federate loggers do not address these needs because of the transmission protocols used by the current generation of RTIs. Other tools to instrument and manage the execution of the federation that are just now coming of age should help the test community. Other tools to help federation design and development such as the OMDT and the FEPW are also useful, and as they become more automated, should reduce the work load during federation development.

There were several software support tools available that JADS used to support the federation development process. More tools are currently being developed and will be made available by DMSO. Open tool interfaces have also been developed (e.g., data interchange formats) to facilitate commercial tool development. In addition, several DoD agencies have ongoing small business innovative research (SBIR) initiatives developing HLA support tools. These tools are designed to provide automated support for development of HLA object models (OMs), planning the federation execution environment, and optimizing the RID file. During the development of the EW Test, JADS used the following tools.

12.5.1 Object Model Development Tool

This tool provides support for developing HLA OMs, generating RTI FED, and exchanging OMs with the object model library (OML). The OMDT automates the process of constructing SOMs and FOMs. The tool provides an HLA user with an interface to OMs consistent with the tabular views defined in the HLA OMT specification. As an OM is constructed, the OMDT performs consistency checking to ensure that the SOM complies with HLA rules. Limitations in the format, organization, and content permitted caused JADS to develop an ICD to fully document all aspects of JADS federation implementation.

In addition to aid in constructing object models, the OMDT provides the user with an interface to DMSO's M&S resource repository. This software allows potential users to browse, download and upload OMs to the repository. After the object model is built using the OMDT, the tool can generate the FED file required by the RTI to execute the federation. The FED file is also used for compliance testing of federates.

JADS used an early version of the OMDT software. Many improvements have been developed and are available from DMSO.

12.5.2 Federation Execution Planner's Workbook

This tool assists with planning the federation execution environment, identifying the hardware and network environment, and specifying a federate's responsibilities for providing and consuming federation data.

Originally, JADS viewed the FEPW as the primary tool for communicating requirements to the RTI development community. JADS began working with DMSO to articulate our perceived requirements for RTI performance in May 1997. While we could articulate what performance we wanted out of the federation, we weren't sure what aspects of performance were critical from an RTI builder's perspective. Our first attempt was to create a system specification to describe RTI/network performance. DMSO proposed using the first generation of RTI performance workbooks which were Excel spreadsheets being designed for users of federations to communicate different aspects of RTI performance requirements. We completed these and have subsequently worked through two other versions of DMSO tools.

While it remains to be seen how well the tools facilitate the communication among RTI developers and federation developers, we found the FEPW tool to be extremely useful in creating the JADS EW Test FOM. We constantly referred to the FEPW, our own concept model spreadsheets, and our ICD. The FOM development became another cross-check of the different design representations prior to federate development and integration.

Throughout the process of developing and articulating FOM requirements, JADS raised questions and provided comments to improve not only the product but also the level of understanding on both the RTI developer and on the federation developer. DMSO is working to couple the OMDT and the FEPW to cut down on data entry duplication. There is still work that needs to be done. For example, one of the critical aspects of RTI performance seems to be the amount of computing resources available for it to use. Yet articulating this requires the developer to quantify the tick rate without necessarily understanding the timing implications to the simulation and to overall performance of the federate.

12.5.3 RTI Initialization Data Editor

The RID file editor tool supports federation development by producing a default RID file or by permitting federation managers to optimize selected elements of the RID file for the specific federation execution. While this tool should be useful, JADS did not use the editor. DMSO led us through the editing process, making the editor unnecessary.

12.6 Verification and Validation

T&E will demand more provable quality of results than other communities. This means that the RTI developers need to use repeatable mature software development processes to develop and maintain the RTI products used by T&E. Acquisition decisions and legal actions can hinge on the results of T&E events. The RTI needs to be as trusted as the other hardware and software components in the test event. Furthermore, the longevity of a T&E federation may well last through the development of a system. This could be well over a decade. The federation will be accredited and that accreditation must remain in force or be continually updated during that time. During that decade of development, change in the architecture is inevitable. However, the change and the cost of re-accreditation must be managed. The T&E community will expect changes to be documented in some fashion similar to version description documents. This is not done in the current generation of RTIs. Shortfalls in documentation and developer discipline will force the T&E community to invest more in tests and tools to ensure RTI performance is up to par.

13.0 Summary

HLA has utility for T&E. It is an enabling technology for distributed testing. As more simulations become HLA compliant, they become resources to the test designer looking to create a richer, more realistic synthetic environment for testing. However, HLA is still maturing. As it matures, T&E has requirements that need to be met.

The first requirement is that HLA be accepted so that simulations become available for T&E. The ultimate acceptance of HLA rests on the availability of suitable RTIs and on the RD&E and T&E communities' ability to create a workable method of reusing simulations within the current rules of HLA. While current RTIs are demonstrably functional, DMSO has not clearly demonstrated a workable method of reuse. Pockets within the RD&E community (and within the T&E community) are advocating RTI interoperability and standard data structures and definitions to help them deal with reuse. While RTI interoperability and standard data structures would improve reusing simulations, neither of these is in the best interest of the T&E community. This observation is discussed in the context of the remaining T&E requirements below.

Standard data structures and definitions are useful to the T&E community, but as JADS found, there are practical reasons why data formats may need to be changed. The JADS ETE Test demonstrated this requirement through the development of multiple interfaces to allow them to live within the constraints of a satellite communications link. T&E will find other constraints as live players are mixed with simulations. T&E require the flexibility to change data formats to meet the needs of the test.

The T&E community also requires RTI performance. Specific performance requirements can't be articulated without articulating the supporting computer/communications hardware and operating system software. In fact, we don't know enough about future tests to even understand all the possible areas of performance that are important to the T&E community. JADS found two: latency, and size of object space. Others exist. In lieu of specific requirements, this observation can be made; the T&E community will be focused on RTI performance more than on RTI services. Commercial competition seems to be the best mechanism for improving performance. However, competition is at odds with RTI interoperability. The RD&E community's desire to make RTIs interoperable may well reduce the incentive for commercial vendors to improve RTI performance beyond today's levels.

The T&E community will be equally focused on making sure that the RTI development processes, testing, and documentation support the VV&A of long-term federations. Poor documentation will increase the cost of long-term ownership of a T&E federation by requiring additional testing to support VV&A. However, testing quality into software is not as effective as designing it in. Good design practices are usually accompanied by good documentation practices. Good documentation will allow the federation owner to understand what risks the RTI may bring to the federation and, as changes occur, what impact the changes may have on the federation. The T&E community will require quality documentation on RTIs that it uses.

HLA needs to continue to develop and evolve to better help the T&E community solve shortfalls in their ability to test new systems. The current rules and tools are not an impediment to the use of HLA by the T&E community. Current RTIs provide the services that T&E require. However, not enough is known about the different facets of performance to state what performance levels T&E will require. Each federation will have to make its own statement of performance. More experience is needed. Meanwhile, each T&E facility and test organization should begin creating a trained cadre of personnel to prepare to use HLA effectively. Familiarity with HLA, object-oriented design, and C++ is necessary to understand where and how HLA may be applied within

your T&E enterprise. Finally, there is no substitute for experience. Formal education only takes you part way up the learning curve. Organizations that expect to use HLA in the future need to practice with the tools now so that they are ready when they need to either develop a federation or à federate. Above all, the T&E community needs to remain involved in HLA to make sure that HLA does not evolve into something that the community can't use.

Appendix A - HLA Terms

affected attributes	The specific attributes of an object class instance whose value in a federation execution may be affected by that instance's participation in a dynamic interaction with another object class.
application programmer's interface (API)	A library of function calls which allows a federate to interact with the runtime infrastructure.
association	A type of static relationship between two or more object classes, apart from class-subclass or part-whole relationships.
attribute	A named portion of an object state.
attribute ownership	The property of a federate that gives it the responsibility to publish values for a particular object attribute.
cancellation	A mechanism used in optimistic synchronization mechanisms such as time warp to delete a previously scheduled event. Cancellation is a mechanism used within the time warp executive and is normally not visible to the federate. It is implemented (in part) using the runtime infrastructure's event retraction mechanism.
causal order	A partial ordering of messages based on the "causally happens before" ($\rightarrow$) relationship. A message delivery service is said to be causally ordered if for any two messages M_1 and M_2 (containing notifications of events E_1 and E_2, respectively) that are delivered to a single federate where $E_1 \rightarrow E_2$, then M_1 is delivered to the federate before M_2.
class	A description of a group of objects with similar properties, common behavior, common relationships, and common semantics.
class hierarchy	A specification of a class-subclass or "is-a" relationship between object classes in a given domain.

conceptual model of the mission space (CMMS)

The conceptual model of the mission space (CMMS) is one of the three components of the DoD technical framework. CMMS is first abstractions of the real world and serves as a frame of reference for simulation development by capturing the basic information about important entities involved in any mission and their key actions and interactions. CMMS is a simulation-neutral view of those entities, actions, and interactions occurring in the real world.

common federation functionality

Agreements on common simulation functionality (services and resources) are finalized among all participants in the federation during the federation development process. Federation members identified during federation design will propose opportunities for common services in areas of assigned responsibilities (also established during federation design) during federation development for discussion and negotiation among all federation participants. For instance, agreements on common representations of terrain (data source, resolution, dynamic versus static, etc.) and environment (required types, data sources, resolution, servers, etc.) would be negotiated and agreed to, as would any relevant federation-specific algorithms (e.g., extrapolation).

component class

An object class that is a component or part of a "composite" object which represents a unified assembly of many different object classes. The identification of a component class in the object model template should include cardinality information.

conceptual analysis

The step in the federation development and execution process which establishes the conceptual framework for the federation. It feeds the design of the overall federation structure. The conceptual view of the objects and interactions that must be represented in the federation is key to identifying reuse opportunities in established federation object models and in determining candidates for federation membership. The high-level representation of the federation scenario refined during conceptual analysis also provides the basis for generation of a more detailed scenario instance during federation design/development.

conservative synchronization

A mechanism that prevents a federate from processing messages out of time stamp order. This is in contrast to *optimistic* synchronization. The Chandy/Misra/Bryant null message protocol is an example of a conservative synchronization mechanism.

constrained simulation	A simulation where time advances are paced to have a specific relationship to wall-clock time. These are commonly referred to as real-time or scaled-real-time simulations. Here, the terms *constrained simulation* and *(scaled) real-time simulation* are used synonymously. Human-in-the-loop (e.g., training exercises) and hardware-in-the-loop (e.g., test and evaluation simulations) are examples of constrained simulations.
coordinated time advancement	A time advancement mechanism where logical clock advances within each federate only occur after some coordination is performed among the federates participating in the execution, e.g., to ensure that the federate never receives an event notice in its past. Aggregate level simulation protocol, for example, uses coordinated time advancement.
current time (of a federate)	Same as federate time.
event	A change of object attribute value, an interaction between objects, an instantiation of a new object, or a deletion of an existing object that is associated with a particular point on the federation time axis. Each event contains a time stamp indicating when it is said to occur (also see definition of message).
event notice	A message containing event information.
exception	An exception in the programming language sense of a possible error - signaling return value. The initiator will be informed of these exceptions.
federate	A member of a high level architecture federation. All applications participating in a federation are called federates. In reality, this may include federate managers, data collectors, live entity surrogates simulations, or passive viewers.
federate time	Scaled wall-clock time or logical time of a federate, whichever is smaller. Federate time is synonymous with the "current time" of the federate. At any instant of an execution different federates will, in general, have different federate times.
federation	A named set of interacting federates, a common federation object model, and supporting runtime infrastructure, that are used as a whole to achieve some specific objective.

federation execution	The federation execution represents the actual operation, over time, of a subset of the federate and the runtime infrastructure initialization data taken from a particular federation. It is the step where the executable code is run to conduct the exercise and produce the data for the measures of effectiveness for the federation execution.
federation execution sponsor	Federation development begins with a user and a requirement. The federation execution sponsor is the organization that uses the results and/or products from a specific application of modeling and simulation. It is the group responsible for establishing the need for the development and execution of a federation. They also establish the framework for the measures of effectiveness by which the results of the execution are employed.
federation object model (FOM)	An identification of the essential classes of objects, object attributes, and object interactions that are supported by a high level architecture federation. In addition, optional classes of additional information may also be specified to achieve a more complete description of the federation structure and/or behavior.
federation objectives	This is the statement of the problem that is to be addressed by the establishment and execution of a federation. The description of the problem domain implicit in the objectives statement is critical for focusing the domain analysis activities in the conceptual analysis phase. It specifies the top-level goals of the federation and may specify the operational need or shortfall from which federation developers will derive a scenario for the federation execution. The federation objectives drive this specification, as the scenario development phase must utilize the statement of the objectives to generate a viable context for system evaluations intrinsic to the federation objectives. High-level testing requirements implied in the federation objectives may also drive the identification of well-defined "test points" during development of the federation scenario.

federation time axis	A totally ordered sequence of values where each value represents an instant of time in the physical system being modeled, and for any two points T_1 and T_2 on the federation time axis, if $T_1 < T_2$, then T_1 represents an instant of physical time that occurs before the instant represented by T_2. Logical time, scaled wall-clock time, and federate time specify points on the federation time axis. The progression of a federate along the federation time axis during an execution may or may not have a direct relationship to the progression of wall-clock time.
fidelity	The similarity, both physical and functional, between the simulation and that which it simulates.
federation required execution details (FRED)	The federation required execution details (FRED) is a global specification of several classes of information needed by the runtime infrastructure to instantiate an execution of the federation. Additional execution-specific information needed to fully establish the "contract" between federation members (e.g., publish responsibilities, subscription requirements, etc.) is also documented in the FRED. The set of management requirements provides one source of input to the FRED specification, which will be recorded in a standardized format.
Greenwich mean time (GMT)	Mean solar time for the Greenwich meridian, counted from midnight through 24 hours. Also called universal time [coordinated] or Zulu time.
happens before, causal ($\rightarrow$)	A relationship between two actions A_1 and A_2 (where an action can be an event, a runtime infrastructure (RTI) message send, or an RTI message receive) defined as follows: (i) if A_1 and A_2 occur in the same federate/RTI, and A_1 precedes A_2 in that federate/RTI, then $A_1 \rightarrow A_2$, (ii) if A_1 is a message send action and A_2 is a receive action for the same message, then $A_1 \rightarrow A_2$, and (iii) if $A_1 \rightarrow A_2$ and $A_2 \rightarrow A_3$, then $A_1 \rightarrow A_3$ (transitivity).
happens before, temporal ($\rightarrow_t$)	A relationship between two events E_1 and E_2 defined as follows: if E_1 has a smaller time stamp than E_2, then $E_1 \rightarrow_t E_2$. The runtime infrastructure provides an internal tie-breaking mechanism to ensure (in effect) that no two events observed by a single federate contain the same time stamp.

independent time advancement	A means of advancing federate time where advances occur without explicit coordination among federates. Distributed interactive simulation is an example of a federation using independent time advancement.
interaction	An explicit action taken by an object, that can optionally (within the bounds of the federation object model) be directed toward other objects, including geographical areas, etc.
interaction parameters	The information associated with an interaction which objects potentially affected by the interaction must receive in order to calculate the effects of that interaction on its current state.
known object	An object is known to a federate if the federate is reflecting or updating any attributes of that object.
lower bound on the time stamp (LBTS)	Lower bound on the time stamp (LBTS) of the next time stamp ordered (TSO) message to be received by a runtime infrastructure (RTI) from another federate. Messages with time stamp less than LBTS are eligible for delivery by the RTI to the federate without compromising time stamp order delivery guarantees. TSO messages with time stamp greater than LBTS are not yet eligible for delivery. LBTS is maintained within the RTI using a conservative synchronization protocol.
local time	The mean solar time for the meridian of the observer.

logical time

A federate's current point on the logical time axis. If the federate's logical time is T, all time stamp ordered (TSO) messages with time stamp less than T have been delivered to the federate, and no TSO messages with time stamp greater than T have been delivered; some, though not necessarily all, TSO messages with time stamp equal to T may also have been delivered. Logical time does not, in general, bear a direct relationship to wall-clock time, and advances in logical time are controlled entirely by the federates and the runtime infrastructure (RTI). Specifically, the federate requests advances in logical time via the time advance request and next event request RTI services, and the RTI notifies the federate when it has advanced logical time explicitly through the time advance grant service, or implicitly by the time stamp of TSO messages that are delivered to the federate. Logical time (along with scaled wall-clock time) is used to determine the current time of the federate (see definition of federate time). Logical time is only relevant to federates using time stamp ordered message delivery and coordinated time advances, and may be ignored (by requesting a time advance to "infinity" at the beginning of the execution) by other federates.

logical time axis

A set of points (instants) on the federation time axis used to specify before and after relationships among events.

look-ahead

A value used to determine the smallest time stamped message using the time stamp ordered service that a federate may generate in the future. If a federate's current time (i.e., federate time) is T, and its look-ahead is L, any message generated by the federate must have a time stamp of at least T+L. In general, look-ahead may be associated with an entire federate (as in the example just described), or at a finer level of detail, e.g., from one federate to another, or for a specific attribute. Any federate using the time stamp ordered message delivery service must specify a look-ahead value.

mean solar time	A time measurement where time is measured by the diurnal motion of a fictitious body (called "mean sun") which is supposed to move uniformly in the celestial equator, completing the circuit in one tropical year. Often termed simply "mean time." The mean sun may be considered as moving in the celestial equator and having a right ascension equal to the mean celestial longitude of the true sun. At any given instant, mean solar time is the hour angle of the mean sun. In civil life, mean solar time is counted from the two branches of the meridian through 12 hours; the hours from the lower branch are marked a.m. (ante meridian), and those from the upper branch, p.m. (post meridian). In astronomical work, mean solar time is counted from the lower branch of the meridian through 24 hours. Naming the meridian of reference is essential to the complete identification of time. The Greenwich meridian is the reference for a worldwide standard of mean solar time called Greenwich mean time or universal time [coordinated].
message	A data unit transmitted between federates containing at most one event. Here, a message typically contains information concerning an event, and is used to notify another federate that the event has occurred. When containing such event information, the message's time stamp is defined as the time stamp of the event to which it corresponds. Here, a "message" corresponds to a single event, however the physical transport media may include several such messages in a single "physical message" that is transmitted through the network.
message (event) delivery	Invocation of the corresponding service (reflect attribute values, receive interaction, instantiate discovered object, or remove object) by the runtime infrastructure to notify a federate of the occurrence of an event.
model	A physical, mathematical, or otherwise logical representation of a system, entity, phenomenon, or process. [DoD 5000.59]
object	A fundamental element of a conceptual representation for a federate that reflects the "real world" at levels of abstraction and resolution appropriate for federate interoperability. For any given value of time, the state of an object is defined as the enumeration of all its attribute values.

object model	A specification of the objects intrinsic to a given system, including a description of the object characteristics (attributes) and a description of the static and dynamic relationships that exist between objects.
object model framework	The rules and terminology used to describe high level architecture object models.
object ownership	Ownership of the identification attribute of an object, initially established by use of the instantiate object interface service. Encompasses the privilege of deleting the object using the delete object service. Can be transferred to another federate using the attribute ownership management services.
optimistic synchronization	A mechanism that uses a recovery mechanism to erase the effects of out-of-order event processing. This is in contrast to *conservative* synchronization. The time warp protocol is an example of an optimistic synchronization mechanism. Messages sent by an optimistic federate that could later be canceled are referred to as optimistic messages.
owned attribute	An object attribute that is explicitly modeled by the owning federate. A federate that owns an attribute has the unique responsibility to provide values for that attribute to the federation, through the runtime infrastructure, as they are produced.
protocol catalog	The protocol catalog is envisioned as an on-line database that will contain standard definitions and formats of data exchanged between distributed simulations. This will help achieve a particular "collective" functionality distributed among multiple federates (e.g., air defense, logistics, etc.). During federation design, this repository is accessed (via automated browsing tools) to identify individual interactions for which a federate will be required, thus helping to define the federation design. The database will be accessible via the World Wide Web. Copies of the protocol catalog can be made and extended by government agencies as necessary to cover classified data. An official unclassified copy will be maintained by the distributed interactive simulation standards workshop.
real time	The actual time in which a physical process occurs.
real-time simulation	Same as constrained simulation.

reflected attribute	An object attribute that is represented but not explicitly modeled in a federate. The reflecting federate accepts new values of the reflected attribute as they are produced by some other federation member and provided to it by the runtime infrastructure.
retraction	An action performed by a federate to unschedule a previously scheduled event. Event retraction is visible to the federate. Unlike "cancellation" that is only relevant to optimistic federates such as time warp, "retraction" is a facility provided to the federate. Retraction is widely used in classical event oriented discrete event simulations to model behaviors such as preemption and interrupts.
runtime infrastructure initialization data (RID)	The data required by the runtime infrastructure for operation. The required data come from two distinct sources, the federation object model product, and the federation required execution details.
runtime infrastructure (RTI)	The general purpose distributed operating system software, which provides the common interface services during the runtime of a high level architecture federation.
scaled wall-clock time	A quantity derived from a wall-clock time defined as offset +[rate*(wall-clock time - time of last exercise start or restart)]. All scaled wall-clock time values represent points on the federation time axis. If the "rate" factor is k, scaled wall-clock time advances at a rate that is k time faster than wall-clock time.

scenario development	In this phase, the federation developer(s) formulate a scenario whose execution and subsequent evaluation will lead toward achieving the study objectives set forth by the federation sponsor. The scenario provides an identification of the major entities that must be represented by the federation, a conceptual description of the capabilities, behavior, and relationships (interactions) between these major entities over time, and a specification of relevant environmental conditions (e.g., terrain, atmospherics, etc.). Initial and termination conditions are also provided. The style and format of the scenario documentation (e.g., graphics, tables, text) are entirely at the discretion of the federation developer. However, communities of use may wish to establish scenario documentation standards among themselves to facilitate reuse of scenario components. The output of this phase is a functional-level scenario description, which is provided as input to the conceptual analysis phase. Certain key activities during conceptual analysis may also drive reiterations of the scenario development phase.
scheduling an event	Invocation of a primitive (update attribute values, send interaction, instantiate object, or delete object) by a federate to notify the runtime infrastructure (RTI) of the occurrence of an event. Scheduling an event normally results in the RTI sending messages to other federates to notify them of the occurrence of the event.
simulation	A method for implementing a model over time. Also, a technique for testing, analysis, or training in which real-world systems are used, or where real-world and conceptual systems are reproduced by a model. [DoD 5000.59]
simulation object model (SOM)	A specification of the intrinsic capabilities that an individual simulation offers to federations. The standard format in which simulation object models are expressed provides a means for federation developers to quickly determine the suitability of simulation systems to assume specific roles within a federation.

time

The measurable aspect of duration. Time makes use of scales based upon the occurrence of periodic events. These are the day, depending on the rotation of the earth; the month, depending on the revolution of the moon around the earth; and the year, depending upon the revolution of the earth around the sun. Time is expressed as a length on a duration scale measured from an index on that scale. For example: 4 p.m. local mean solar time means that 4 mean solar hours have elapsed since the mean sun was on the meridian of the observer.

time flow mechanism

The approach used locally by an individual federate to perform time advancement. Commonly used time flow mechanisms include event driven (or event stepped), time driven, and independent time advance (real-time synchronization) mechanisms.

time management

A collection of mechanisms and services to control the advancement of time within each federate during an execution in a way that is consistent with federation requirements for message ordering and delivery.

time stamp (of an event)

A value representing a point on the federation time axis that is assigned to an event to indicate when that event is said to occur. Certain message ordering services are based on this time stamp value. In constrained simulations, the time stamp may be viewed as a deadline indicating the latest time at which the message notifying the federate of the event may be processed.

time stamp order (TSO)

A total ordering of messages based on the "temporally happens before" ($\rightarrow_t$) relationship. A message delivery service is said to be time stamp ordered if for any two messages M_1 and M_2 (containing notifications of events E_1 and E_2, respectively) that are delivered to a single federate where $E_1 \rightarrow_t E_2$, then M_1 is delivered to the federate before M_2. The runtime infrastructure (RTI) ensures that any two time stamp order messages will be delivered to all federates receiving both messages in the same relative order. To ensure this, the RTI uses a consistent tie-breaking mechanism to ensure that all federates perceive the same ordering of events containing the same time stamp. Further, the tie-breaking mechanism is deterministic, meaning repeated executions of the federation will yield the same relative ordering of these events if the same initial conditions and inputs are used, and all messages are transmitted using time stamp ordering.

transportation service	A runtime infrastructure provided service for transmitting messages between federates. Different categories of service are defined with different characteristics regarding reliability of delivery and message ordering.
true global time	A federation-standard representation of time synchronized to Greenwich mean time or universal time [coordinated] (as defined in this glossary) with or without some offset (positive or negative) applied.
unconstrained simulation	A simulation where there is no explicit relationship between wall-clock time and the rate of time advancements. These are sometimes called "as-fast-as-possible" simulations, and these two terms are used synonymously here. Analytic simulation models and many constructive "war game" simulations are often unconstrained simulations.
universal time [coordinated] (UTC)	The same as Greenwich mean time. A nonuniform time based on the rotation of the earth, which is not constant. Usually spoken as coordinated universal time.
wall-clock time	A federate's measurement of true global time, where the measurement is typically output from a hardware clock. The error in this measurement can be expressed as an algebraic residual between wall-clock time and true global time or as an amount of estimation uncertainty associated with the wall-clock time measurement software and the hardware clock errors.

Appendix B - DoD HLA Directive

Under Secretary of Defense
(Acquisition and Technology)

Sept. 10, 1996

MEMORANDUM FOR: SECRETARIES OF THE MILITARY DEPARTMENTS
CHAIRMAN OF THE JOINT CHIEFS OF STAFF
UNDER SECRETARIES OF DEFENSE
ASSISTANT SECRETARIES OF DEFENSE
GENERAL COUNCIL OF THE DEPARTMENT OF DEFENSE
INSPECTOR GENERAL OF THE DEPARTMENT OF DEFENSE
DIRECTOR, OPERATIONAL TEST AND EVALUATION
ASSISTANTS TO THE SECRETARY OF DEFENSE
DIRECTOR OF ADMINISTRATION AND MANAGEMENT
DIRECTORS OF THE DEFENSE AGENCIES

SUBJECT: DoD High Level Architecture (HLA) for Simulations

References: DoD Directive 5000.59, "DoD Modeling and Simulation (M&S) Management," January 4, 1994
DoD 5000.59-P, "DoD Modeling and Simulation Master Plan (MSMP)," October 1995

Under the authority of reference (a), and as prescribed by reference (b), I designate the High Level Architecture as the standard technical architecture for all DoD simulations.

The baseline HLA is defined by three inter-related elements: HLA Rules Version 1.0 (v.1.0), HLA Interface Specification v.1.0, and HLA Object Model Template v.1.0. The evolution of the HLA will be managed by the DoD Executive Council for Modeling and Simulation (EXCIMS) through its Architecture Management Group (AMG). This structure provides a means for the DoD Components to identify and address any emergent issues in subsequent refinements to the HLA. Compliance with the HLA does not mandate the use of any particular implementation of supporting software such as the Runtime Infrastructure.

DoD Components shall review all of their simulation projects and programs by the second quarter fiscal year (FY) 1997 in order to establish plans for near-term compliance with the HLA. The Department shall cease further development or modification of all simulations which have not achieved, or are not in the process of achieving, HLA-compliance by the first day of FY 1999, and shall retire any non-compliant simulations by the first day of FY 2001. EXCIMS is to monitor progress and advise me if any emergent events affect their viability.

To monitor compliance with the HLA, the DoD Components shall submit an initial report to the Defense Modeling and Simulation Office (DMSO) by June 30, 1997, which summarizes their HLA-compliance intentions for each simulation the Component owns or sponsors, organized into three categories:

- HLA-compliance actions initiated immediately
- HLA-compliance actions initiated at a specified future date
- no HLA compliance planned (thus requiring eventual retirement or a waiver)
-

The DoD Components shall submit periodic updates to these initial reports as required to ensure their accuracy and completeness. DMSO shall establish a mechanism to provide for formal certification of compliance and shall provide me with periodic reports on the Department's progress towards compliance with the HLA.

If a Component believes it is impractical for a simulation to comply with the HLA, or that HLA-compliance cannot be achieved in a timely manner, it may submit a waiver request to the Director of Defense Research and Engineering, the Chair of the EXCIMS. In consultation with the EXCIMS and its Training, Analysis, and Acquisition Councils, I will then decide if an exception to the HLA-compliance requirement is warranted, and if so, the form of that exception.

This mandate for HLA-compliance supersedes all previous requirements for DoD simulations to comply with other simulation standards such as Distributed Interactive Simulation or Aggregate-Level Simulation Protocol. It is expected that new industry standards to support the HLA will emerge. In consultation with the EXCIMS and its AMG, I will evaluate the suitability of such standards for the Department as they are established.

The DoD point of contact for the HLA is the Defense Modeling and Simulation Office at (703) 998-0660 or hla@dmso.mil. The HLA documents are available at http://www.dmso.mil/.

\original signed\
Paul G. Kaminski

Appendix C – Acronyms and Definitions

A/C	aircraft
ACETEF	Air Combat Environment Test and Evaluation Facility, Patuxent River, Maryland; Navy facility
ADRS	Automated Data Reduction Software
ADS	advanced distributed simulation
AFEWES	Air Force Electronic Warfare Evaluation Simulator, Fort Worth, Texas
AFOTEC	Air Force Operational Test and Evaluation Center, Kirtland Air Force Base, New Mexico
ALSP	aggregate level simulation protocol
ANIU	air network interface unit
API	application program interface
ARPA	Advanced Research Projects Agency
C4ISR	command, control, communications, computers, intelligence, surveillance and reconnaissance
CMMS	conceptual model of the mission space
CORBA	Common Object Request Broker Architecture
CPU	central processing unit
DDM	data distribution management
DIS	distributed interactive simulation
DMSO	Defense Modeling and Simulation Organization, Alexandria, Virginia
DoD	Department of Defense
DSM	digital system model
DT&E	developmental test and evaluation
ENV	environment
ESPDU	entity state protocol data unit
ETE	JADS End-to-End Test
EW	electronic warfare; JADS Electronic Warfare Test
FED	federation execution data
FEDEP	federation development and execution process
FEPW	Federation Execution Planner's Workbook
FOM	federation object model
FRED	federation required execution details
GMT	Greenwich mean time
GNIU	ground network interface unit
GPS	global positioning system
GSM	ground station module
HITL	hardware-in-the-loop
HLA	high level architecture
Hz	hertz
I/F	interface
IADS	Integrated Air Defense System
ICD	interface control document

ID	identification
IEEE	Institute of Electrical and Electronics Engineers
IP	initial point; Internet protocol
IRIG	Inter-Range Instrumentation Group
IRIX	operating system for the Silicon Graphics, Inc.
JADS	Joint Advanced Distributed Simulation, Albuquerque, New Mexico
JETS	JammEr Techniques Simulator
Joint STARS	Joint Surveillance Target Attack Radar System
JT&E	joint test and evaluation
JTC	jammer technique command
JTF	joint test force
km	kilometer
LAN	local area network
LBTS	lower bound on the time stamp
LGSM	light ground station module
LHC	link health check
LRC	local runtime infrastructure component
M&S	modeling and simulation
ms	milliseconds
NIU	network interface unit
OAR	open air range
OFP	operational flight program
OM	object model
OMDT	object model development tool
OML	object model library
OMT	object model template
OSD	Office of the Secretary of Defense
OT&E	operational test and evaluation
PC	personal computer
PDU	protocol data unit
Pent	pentium
RD&E	research, development, and engineering
RELDISTR	reliable distributor
RF	radio frequency
RFENV	radio frequency environment
RID	RTI initialization data
RPR	real-time platform reference
RTC	reference test condition
RTI	runtime infrastructure
RTIexec	runtime infrastructure executive
RTU	receiver track update
SATCOM	satellite communication
SBIR	small business innovative research
sec	second
SGI	Silicon Graphics, Inc.

SIMNET	simulator network
SISO	Simulation Interoperability Standards Organization
SIT	JADS System Integration Test
SOM	simulation object model
SPJ	self-protection jammer
SUT	system under test
T&E	test and evaluation
T-1	digital carrier used to transmit a formatted digital signal at 1.544 megabits per second
TAMS	Tactical Air Mission Simulator
TCAC	test control and analysis center
TCF	test control federate
TCP	transmission control protocol
TSO	time stamp ordered
TSPI	time-space-position information
TTL	time to live
UDP	user datagram protocol
UTC	universal time (coordinated)
VSTARS	virtual surveillance target attack radar system
VV&A	verification, validation and accreditation
WAN	wide area network